Vorwort

Anfang Juli 2003 fand an der Universität Marburg ein Interdisziplinäres Symposium über grundlegende Orientierungsfragen der Informatik - mit Schwerpunkt Objektorientierung und Grundlagen der Softwaretechnik - statt.

Mit diesem Symposium wollten die Veranstalter an die Grundlagenarbeit anknüpfen, die in den letzten Jahren u.a. in verschiedenen Arbeitskreisen und Workshops zur Terminologie der Softwaretechnik, zur Objektorientierung, zur Theorie der Informatik und zu Wissenschaftstheorie und Wirtschaftsinformatik geleistet wurde. Ein aktueller Anknüpfungspunkt war der Workshop *Erkenntnistheorie - Semiotik - Ontologie (ESO): Die Bedeutung philosophischer Disziplinen für die Softwaretechnik",* der im Rahmen der GI-Tagung 2001 in Wien den Dialog zwischen Informatik und Philosophie wieder aufnahm und fortführte.

Im Aufruf zur Tagung waren folgende Themen genannt worden:

- Hat die Softwaretechnik schon gesicherte Grundlagen und Begriffsbildungen, gibt es so etwas wie eine "Ontologie der Softwaretechnik"?
- Ist (und bleibt) die Objektorientierung ein unbestrittenes Paradigma - spiegelt sie die "natürliche" Weltsicht für Entwickler und Anwender wider? Ist Objektorientierung als methodischer Ansatz für alle Phasen der Software-Entwicklung (von der Analyse bis zur Nutzung der Software) in gleicher Weise angemessen?
- Kommen die involvierten "Subjekte" (Anwender, Entwickler und Betroffene) bei einer solchen Betrachtungsweise zu ihrem Recht?
- Bestimmen die Objekte unsere Handlungen, oder sind es umgekehrt unsere Handlungen, welche die Objekte bestimmen? Welche Rolle können und sollen empirische Ansätze zur Software-Herstellung spielen?
- Wie sind Objektorientierung oder der Ontologie-Ansatz der Informatiker mit den Weltsichten der Philosophen und ihren tradierten Schulen verträglich? Wo liegen aus praktischer bzw. grundsätzlicher Sicht die Stärken und Schwächen der einzelnen Ansätze?
- Woran kann man das Ge- (oder Miss-)lingen von Informatik-Projekten messen, die sich die Unterstützung oder Veränderung von Prozessen oder Vorgängen unserer Lebenswelt zum Ziel gesetzt haben? Wie ist das Verhältnis zwischen Aufwand und Nutzen?

Die Rückmeldungen zu diesem Aufruf waren erfreulich zahlreich: es wurden 14 Beiträge eingereicht (davon 11 aus der Informatik, 2 aus der Philosophie und einer aus den Medienwissenschaften) und einem zweifachen Beurteilungszyklus unterzogen. Als Ergebnis wurden 11 Beiträge ausgewählt, beim Symposium vorgestellt und diskutiert. Dazu kommen zwei eingeladene Vorträge, die das Tagungsthema aus Sicht der Philosophie beleuchten.

Die im folgenden dokumentierten Beiträge lassen sich grob drei Themenbereichen zuordnen.

- Systemgestaltung, Modellierung, Sprache und Semantik:

Zu diesem ersten Bereich rechne ich die Arbeiten von Frieder Nake über die Subjekt-/Objektbeziehung bei der Systemgestaltung, von Ulrich Frank über spezielle Probleme bei der objektorientierten Modellierung, von Sabrina Geißler über Suchmaschinen und die Verarbeitung von Metadaten sowie von Johannes Busse zur Ontologie-Konstruktion.

- Grundlagen und philosophische Bezüge der Softwaretechnik und der Informationssysteme:

Dieser Themenbereich wurde durch den eingeladenen Vortrag von Peter Janich über interdisziplinäre Bezüge zwischen Informatik und Philosophie eingeleitet. Morteza Ghasempour führte in einem weiteren eingeladenen Vortrag durch verschiedene Objekt-, Objektivitäts- und Wahrheitskonzepte im Verlaufe der Philosophiegeschichte. Weitere Beiträge von Bettina Müller, Wolfgang Hesse & Hubert v. Braun und von Roland Kaschek befassen sich mit der Deutung und Bedeutung des Objektbegriffs in der Softwaretechnik und speziell bei der objektorientierten Analyse und der Entwicklung von Informationssystemen.

- Perspektiven der Informatik

In diesem letzten Themenbereich wurden weitere Grundlagenarbeiten vorgestellt, welche die Informatik insgesamt als Wissenschaft und als Profession zum Gegenstand haben. Dies sind die Arbeiten von Ludger Eversmann zu den wissenschaftstheoretischen Perspektiven von Informatik und Wirtschaftsinformatik, von Dirk Siefkes zum interkulturellen und interdisziplinären Charakter der Informatik sowie von Peter Bittner zum Selbstverständnis der Informatik als Profession.

Ich danke allen Autoren, den Tagungseilnehmern, den Mitgliedern des Programmkomitees und allen Mithelfern bei der Tagung für ihre Unterstützung und aktive Mitarbeit am Symposium. Ohne Ihre Mithilfe wäre eine solch gelungene und fruchtbare Veranstaltung nicht möglich gewesen.

Marburg, im Juli 2003

Wolfgang Hesse

Subjekt & Objekt
Participatory Design & Object-oriented Design. Eine Reflexion

Frieder Nake, Universität Bremen

Dem Anlass dieses Aufsatzes entsprechend, will sich der Autor einen Stil genehmigen, wie er im wissenschaftlichen Rahmen nicht immer gern gesehen sein mag. Warum auch, so frage ich mich als einer, der das Glück hat, oft zwischen den Welten zu wandern – warum auch sollte, wenn es um *Perspektiven der Informatik* geht, nicht ein Quentchen essayistischer Stilistik zugelassen sein?

Wolfgang Hesse hatte, seiner Neigung zum Philosophischen folgend, nach solchen Perspektiven der *Informatik* gefragt, die objekt-, subjekt- oder handlungsorientiert genannt werden könnten. Besehen wir die Frage etwas enger, so werden daraus wohl doch Perspektiven der *Software-Entwicklung* werden. Ich will deswegen in diesem Beitrag zunächst kurz sagen, wie ich das Verhältnis beider zueinander sehe.

Um der Perspektive des *Subjekts* zu ihrem Recht zu verhelfen, möchte ich einen – naturgemäß subjektiv gefärbten – Blick auf die aus Skandinavien gekommene partizipative Software-Entwicklung werfen. Davor gibt es eine Seitenbemerkung zum Verhältnis von Entwicklung und Design, die nicht nur der Sprachregelung, sondern auch des Inhalts wegen notwendig ist. Die Perspektive des *Objektes* kommt dann – in objektivistischem Gestus – zu ihrem Recht in der Behauptung einer Notwendigkeit objektorientierter Programmiersprachen.

Eine skeptische Betrachtung zur Art der Objekte, die die Informatik hervorbringt, soll sich anschließen. Ich werde die Objekte als *Zeichen* einer besonderen Art kennzeichnen, die ich explizieren werde. Es trifft sich, dass, was bei Christiane Floyd und Michaela Reisin „autooperationale Form” genannt wird, hier als „algorithmisches Zeichen” erscheint. Auch bei Jay David Bolter finden wir es bereits im Jahre 1991[1].

Die *Handlung* als die dritte hier nachgefragte Perspektive der Informatik spare ich aus. Nicht, dass sie mir unwichtig wäre. Im Gegenteil: sie muss mir gar nicht wichtig werden, da sie es immer ist. Wie sollten Informatiker und Informatikerinnen ihr Geschäft, die Kopfarbeit zu maschinisieren, betreiben, wenn sie nicht auf die Handlungen der Arbeitenden – allgemeiner: der Tätigen – achteten? Die Hessesche Frage will uns also zweifellos vor die Alternative stellen, statt des Objekts oder Subjekts die Handlung selbst ins Zentrum der Anstrengungen um Begriffe, Techniken und Methoden zu stellen.

Das dürfte auf folgendem Schema aufsetzen. Der tätige Mensch existiert, indem er als arbeitendes *Subjekt* verändernd auf ein bearbeitetes *Objekt* einwirkt und indem dieses Verhältnis eine *Handlung* darstellt. Die Handlung erscheint dabei als die mittlere Ebene einer arbeitspsychologischen Differenzierung im ökonomischen Begriff der Arbeit[2]. Arbeit geschieht danach als eine *Tätigkeit*, die umfassend auf ein Bündel von Zielen ausgerichtet ist und den Menschen über einen ausgedehnten Zeitraum hin gefangen nimmt. Solche umfassende Tätigkeit findet als Struktur von *Handlungen* statt, die hinter- oder nebeneinander ausgeführt werden und von denen jede ein der Tätigkeit untergeordnetes Teil-Ziel besitzt, das seinen weiteren Sinn erst aus dem Ziel der Tätigkeit selbst gewinnt. Jede Handlung wiederum besteht aus einzelnen *Operationen*, den kleinsten unterschiedenen Aktivitäts-Bestandteilen.

Die hier angesprochene Sinn-Einheit der Handlung mag das Stichwort für Wolfgang Hesses dritte Alternative gegeben haben. In Subjekt und Objekt erscheinen Operationen und Handlungen. Ohne Subjekt- und Objekt-Bezug wäre Handlung nicht zu fassen. Sie ist deutlich von relationaler Art, während Subjekt und Objekt (naiv gesehen) durchaus als Einzelinstanzen, als Gegen*stände*, gedacht werden können.

[1] Dirk Siefkes weist mich darauf hin, dass seine *hybriden Objekte* ebenfalls das Gleiche meinen.

[2] Die Differenzierung besteht in der Schichtung von umfassender *Tätigkeit*, aktuell betriebener *Handlung* und kontextarmer *Operation* im Detail. Die *Activity Theory* geht solcher Differenzierung nach. Walter Volpert, Yngve Engeström, Arne Raeithel, Susanne Bødker, Bonnie Nardi u.v.m. haben dazu im Gefolge der russischen Pioniere Wygotski, Leontjew, Luria in den letzten Dekaden kräftige Impulse gesetzt. Die skandinavische und Teile der US-Szene der Software-Entwicklung sind geprägt hiervon (s. Raeithel und Volpert in [Coy et al. 92]).

1 Informatik und Software-Entwicklung

Wenden wir uns dem Verhältnis von Informatik und Software-Entwicklung zu. Naiv stehen sie wie der prinzipielle und der ökonomische Zugang zu einem Phänomen zu einander.

Das Phänomen ist das der Existenz von Computern, Netzen, Datenverarbeitung, algorithmischen Modellen, kurz: einer technischen Infrastruktur, die die Grundlagen der Industriegesellschaft so heftig durchzogen und umgewälzt hat, dass ihr bis hin zum Begriff ihrer selbst alles abhanden gekommen ist, was sie ausmachte. Sie nennt sich selbst gern Informationsgesellschaft und hat zur Verstärkung dieser Entwicklung die Wissenschaftsdisziplin Informatik kreiert und ein Bündel von beruflichen Tätigkeiten der Software-Technik hervorgebracht.

Informatik ist in dieser Sichtweise die wissenschaftliche Beschäftigung mit der Konstruktion von Software.[3] Ihre Orte sind die Universität, die Konferenz, die Zeitschrift. Man betreibt sie in eher theoretischer Absicht, will Prinzipien solcher technischen Artefakte, ihrer Schaffung und ihrer Verwendung identifizieren und sie so fassen, dass ihre Befolgung Auswirkungen auf die gesellschaftliche Praxis erlangen kann. Die Informatik bezieht sich ohne Frage auf die Maschinisierung von Kopfarbeit, was zu erläutern hier der Platz fehlt.[4]

Software-Entwicklung hingegen erscheint als jener Bereich gesellschaftlicher Praxis, der der Hervorbringung von Software unter ökonomischen Schranken gewidmet ist und Verwertungsbedingungen beachten muss. Ihre Orte sind Industrieunternehmen, Start-up Firma, HighTec Schmiede, die Messe. Wo der Informatiker rückhaltlos fragen darf und soll, ist es der Software-Entwicklerin bei Androhung von Konventional-Strafen untersagt, Termin- und Mittelschranken zu überschreiten.[5]

Selbstverständlich zeichne ich ein idealisiertes Bild. Es soll der raschen Verständigung über Begriffe dienen. Gerade im Feld der Software ist es faszinierend zu registrieren, in welch hohem Maße theoretische Fragestellungen unmittelbar aus praktischer Not entstehen, und theoretische Ergebnisse direkt in Techniken und Produkte überführt werden. Karl Marx würde vom Zustand der Wissenschaft als *unmittelbarer* Produktivkraft sprechen und damit das Verhältnis von Informatik und Software-Entwicklung auf den Begriff bringen.

Nur erwähnt sei, dass der Begriff Software als Ansammlung von Programmteilen und deren Strukturen, von Daten und ihren Strukturen, von grafischen Präsentationen und Interaktionsstrukturen den Begriff des *Systems* evoziert. Damit muss heute die Dialektik von System und Umwelt gemeint sein, eine Perspektive, der ich hier nicht nachgehen kann.

[3] Man mag hierfür die sehr anregende Sammlung [Schefe et al. 93] zur Informatik als Disziplin konsultieren.

[4] Wer nachlesen möchte, sei auf [Nake 92] aufmerksam gemacht.

[5] Die Sammlung [Desel 01] enthält auch Aufsätze zum Verhältnis von Informatik und Software-Erstellung.

2 Entwicklung, Design, Gestaltung

Man spricht von der *Entwicklung* von Software und meint damit einen Prozess, der Beschreibungen ineinander überführt[6]. An seinem Anfang steht eine Beschreibung, die man oft „Anforderungsdefinition" nennt. Gern wird diese erste Vorgabe als Szenario formuliert. Am Ende des Prozesses steht eine Beschreibung, die „Programm", „Implementierung", „ausführbarer Code" o.ä. genannt wird. Hat die erste Beschreibung reine Textgestalt (mit Skizzen, Diagrammen, Tabellen etc.), so ist die letzte Text in der Form maschineller Lesbarkeit.

Wie jeder und jede weiß, sind die beiden Pole der Kette vom Menschen wie vom Computer lesbar, jedoch ergeben sie für diese zwei Interpreten – wenn solch gemeinsame Benennung gestattet ist – unterschiedlichen, wenngleich verwandten *Sinn*. Ich beeile mich zu versichern, dass es Sinn für den Computer gar nicht geben kann, dass er „Sinn"[7] aber durchaus feststellen mag – ja, dazu gezwungen ist, dies zu tun, damit die erwähnte Beschreibung tatsächlich zu dem führen kann, zu dem sie führen soll – nämlich zu einer Operationenfolge.

Die Entwicklung von Software ist also die Ent*wickelung* einer Beschreibung, die zu Beginn höchsten Grad an Lesbarkeit für den Menschen, jedoch niedrigsten für die Maschine besitzt. Am Ende dieser Entwickelung verhält es sich gerade umgekehrt. Der Mensch, durchaus in der Lage, das Programm zu lesen, tut dies nicht allzu gern und ist häufig noch nicht einmal geschult, es zu tun. Die Maschine hingegen fühlt sich nun – ich anthropomorphisiere aufs Schamloseste – geradezu in ihrem eigentlichen Element. Denn wenn sie die anfänglichen Beschreibungen zu kaum etwas anderem nutzen konnte als festzustellen, dass es sich um eine Kette von Zeichen handelte, so gewinnt sie nun aus dem Programmtext Anlässe dafür, effektiv etwas zu tun, so richtig als Maschine also in Erscheinung zu treten[8].

„Lesbarkeit" bedeutet hierbei die Möglichkeit einer Sinnzuschreibung. Zeichentheoretisch betrachtet, ist solch eine *Zu*schreibung von Sinn aus Anlass einer *Be*schreibung nichts anderes als jener Vorgang, durch den einem *Repräsentamen* (einem Signalangebot), das (sichtbar) wahrgenommen wird, ein *Objekt* und diesen beiden zusammen ein *Interpretant* zugeordnet wird.[9] Die gegebene Beschreibung führt zur Erzeugung einer neuen, die den enthaltenen Sinn (Objekt) und seine Bedeutung (Interpretant) expliziert.

Kurz gesagt, ist das Objekt bei Peirce das, was *bezeichnet* wird. Der Interpretant ist das, was *bedeutet* wird. So jedenfalls will ich es mit Max Bense halten. Über die Wirkungsweise jedes Zeichens – eines steht für ein anderes – hinaus eröffnet diese dreistellige Begrifflichkeit eine Differenzierung, die die Einbettung in eine Kultur von den individuellen Besonderheiten des lebendigen Handelnden zu trennen erlaubt. Die Be*zeich*nung nämlich besitzt relative Allgemeingültigkeit für eine Gruppe oder Gemeinschaft oder Kultur. Die Be*deut*ung dagegen kann nur vom anwesenden Interpreten, von demjenigen also hergestellt werden, der jetzt und hier einem Signalangebot (bei Peirce: Repräsentamen) ausgesetzt ist, es wahrnimmt und durch eine Bedeutungszumessung das Zeichen erst zum vollen Zeichen macht.

[6] Aus der reichhaltigen Literatur zur Software-Entwicklung sei nur an zwei wichtige Äußerungen vom Rande her erinnert: [Brooks 75, DeMarco & Lister 91].
[7] Sinn und „Sinn" (mit Anführungszeichen) markieren verschiedene Begriffe.
[8] Ob das effektive maschinelle Operieren nach menschlichem Maße auch effizient ist, ist eine andere Frage.
[9] Ich verwende die semiotische Terminologie von Charles Sanders Peirce [Peirce 93].

Die drei Begriffe der Überschrift dieses Abschnitts machen darauf aufmerksam, dass die Arbeit des Spezifizierens, Entwerfens, Implementierens, Testens von Software nicht immer „Entwicklung", sondern oft „Design" genannt wird[10]. Im Deutschen auch: „Gestaltung".

Klingt im Wort „Design" im Englischen eine sehr breite Bedeutung an, die nicht mit dem deutschen „Gestaltung" gleichzusetzen ist, so ist doch die Nähe zum Entwerfen als einem kontext- und situationsbasierten, in Alternativen denkenden und nicht auf beste Lösungen zielenden Prozess unverkennbar (vgl. [Floyd 94]). Design zielt auf Artefakte, die eher zu bewerten, zu deuten, dem Geschmacksurteil auf Erfahrungsgrundlage zu unterwerfen sind. Entwicklung dagegen zielt auf Artefakte, die zu bemessen, zu begreifen, dem Korrektheitsurteil auf formaler Grundlage zu unterwerfen sind. Design geht von der Dialektik der Situation aus, die durch Design verändert wird, Entwicklung von ihrer Optimierung.

Design bettet die Artefakte, die erst entstehen, in Sinnzusammenhänge ein, sieht Software-Entwicklung als einen situierten Prozess wechselseitigen Lernens in veränderlichen, offenen Kontexten. Systeme sind bestenfalls Approximationen, während sie dem entwickelnden Ingenieur als Objekte ständiger Vervollkommnung erscheinen.

Für Gestaltung, die der ästhetischen Dimension, der Schönheit, dem Schein und der Kunst einen Schritt näher kommt, gilt alles Gesagte um so entschiedener. Die Gestalt ist die aktuale, in einer Art von Balance befindliche äußere Form einer inneren Vorstellung. Sie ist die schwebende Dialektik von Vorstellung und Herstellung. Gestaltung zielt auf sie.[11]

3 Partizipative Software-Entwicklung

Zu Beginn der 80er Jahre griffen Anwendungen des Computers über die industrielle Arbeit und die Großverwaltung hinaus; die Informationstechnik tauchte im Alltag des Büros auf. Bald war deutlich, dass die informellen Kontexte nicht abstrakt am Programmiertisch in Software verwandelt werden konnten.

Die Maschinisierung von Kopfarbeit verlangte plötzlich danach, die sog. Betroffenen einzubeziehen. Das sind mögliche spätere Benutzende eines erst in Entwicklung befindlichen Software-Systems. In Pilotprojekten, angesteckt von Beispielen wie dem skandinavischen UTOPIA Projekt, wurde Software *partizipativ* entwickelt. Kommende Benutzende wurden vor allem in frühen Phasen der Entwicklung sowie vor Inbetriebnahme gehört. In Teams wurden neuartige Entwurfsmethoden erprobt. Formales, auf sicher scheinendem Boden stehendes Vorgehen wich einem interpretierenden und konfliktreichen Aushandeln und Probieren. Für das traditionelle Konstruieren technischer Systeme ein Affront, der oft mit Kopfschütteln und Unverstand abgetan wurde (und noch immer wird?).

Die Skandinavier öffneten einen Weg, der vermutlich größere Nachhaltigkeit aufweist als der Streit um die richtige Abfolge der Phasen der Software-Entwicklung erzielen konnte.[12] Denn das Prinzip der partizipativen Entwicklung von Systemen für Büro oder Arztpraxis führte im Grunde die ästhetische Dimension in die Gestaltung von Software ein. Sie wurde zum nur gering kontrollierbaren, dennoch aber technisch bestimmten Artefakt, das zwar zu-

[10] Hierzu gibt es im englischen Sprachraum eine Explosion an Literatur, auch an reflektierender. Ich verweise aus zufälliger Vorliebe auf [Dasgupta 91, Floyd et al. 92, Winograd 96]. Eine Brücke zur skandinavischen Tradition schlagen [Ehn & Malmborg 98]
[11] Arno Rolf wird nicht müde, die Informatik als eine Gestaltungswissenschaft zu begreifen. Unrecht hat er nicht.
[12] S. zusammenfassend knapp etwa [Newman & Lamming 95] und breiter [Greenbaum & Kyng 91], kritisch [Whitaker et al. 91].

nächst noch ganz der Arbeitswelt verhaftet blieb, bald aber den Sprung hinaus in die gesamte Gesellschaftlichkeit, in die Kultur tat.[13]

Spätestens jetzt war der traditionelle Zugang zu dieser Art technischer Artefakte verbaut. Die mediale Zeit der Software brach an, die Zeit, die schließlich zur *Open Source* Bewegung führte. Eine unvorstellbare Tatsache, dass Software über Kontinente und lange Zeiten hinweg von riesenhaften, wechselnden Teams mit Erfolg und von manchmal besserer Qualität entwickelt wurde als von Professionellen. Von Teams, die nach klassischer Vorstellung gar keine waren, da sie sich weder kannten, noch zusammen arbeiteten, noch gemanaget wurden.

Nicht dass Professionalität ihre Bedeutung generell verloren hätte! Es gibt stets Bereiche, wo ihr Vorgehen günstiger ist als das der Open Source Bewegung. Was aber in die Welt gekommen ist und im partizipativen Design[14] der 80er Jahre seine Wurzeln besitzt, ist ein anarchistisch-demokratisches Element hohen Engagements, das eine noch unbekannte Sprengkraft besitzen könnte. Mit entsprechenden Argus-Augen wird mit Sicherheit beobachtet, was sich da am Rande der Höhle des Löwen tut.

So unfassbar die ästhetische Dimension uns oft erscheint, so unfassbar ist auch das Phänomen, von dem die Open Source Bewegung abhängt: die stillschweigende Übereinkunft, sich aus konvergierendem Interesse heraus an Prinzipien zu halten, die nicht per Sanktion eingeklagt werden können. Die Jagd nach der Bewältigung der Komplexität findet nicht statt. Die Beteiligten bilden rhizomartige Geflechte und spotten der Wurzelbäume.

4 Objektorientierung als Prinzip

Kaum jemand, der nicht seit mehreren Jahren – kommt es zur Frage nach der bevorzugten Methode der Software-Entwicklung – von der Objekt-Orientierung schwärmte. Deren für bestimmte Situationen günstige Möglichkeiten werden bei Verallgemeinerung zu einer Methode tendenziell der notwendigen Betrachtung von Situation und Kontext entzogen.[15]

Dabei können wir – wenigstens für die Programmiersprachen – behaupten, dass der Orientierung am Objekt eine logische Zwangsläufigkeit innewohnt. Das sei kurz ausgeführt.

Wir gehen davon aus, dass die Funktion eines Computers sich aus einer Folge von Basisschritten ergibt. Jeder Schritt ist die Anwendung einer Operation auf ein Objekt. Objekt und Operation können komplex strukturiert sein. Die lineare Folge der Schritte wird aktual aus dem Lauf eines nichtlinear notierten Programms gewonnen, das auf Eingabedaten angewandt wird. Programmierung besteht darin, in Programmen Kontrollstrukturen als Steuerungen zu schreiben. Die Beschreibung wird mit Mitteln einer Programmiersprache ausgedrückt.

Programmiersprachen müssen folglich Möglichkeiten eröffnen, Objekte und Operationen sowie Abläufe zu beschreiben. Zwei Extreme werden dabei möglich. Erstens der Verzicht auf Reichhaltigkeit des Ausdrucks bei den Objekten. Zu nur einer Sorte von Objekten (alles wird Liste!) werden Operationen reichhaltig definiert: das ist funktionale Programmierung; Lisp ist ihr paradigmatischer Fall.

Das andere Extrem verzichtet auf Reichhaltigkeit bei den Operationen. Zu nur einer Art von Operation (alles wird Nachrichtenaustausch!) werden Objekte reichhaltig definierbar: das ist objektorientierte Programmierung; Smalltalk ist ihr paradigmatischer Fall.

[13] In der BRD brachte diese Bewegung die Software-Ergonomie hervor, den Versuch, mit traditionellen Mitteln zu reagieren, unbewusst sich gegen das Eindringen der Ästhetik zu wehren. Das ist gescheitert, hat aber gewirkt.

[14] Hier sei zur Zweisprachigkeit des Aufsatz-Titels vermerkt, dass in seinem deutschen Teil die Reflexion und Philosophie, im amerikanischen hingegen die Tat und Technik anklingen sollen. Auch die freiwillige Aufgabe der eigenen Sprache, die die Deutschen so sehr pflegen.

[15] Ohne den kritischen Seitenhieb siehe [Züllighoven 98] als eine nicht dogmatische Einführung.

Das Zwangsläufige der Extreme liegt darin, dass sie in der angedeuteten Vorstellung vom Programmieren enthalten sind. Die meisten verbreiteten Sprachen liegen, als prozedurale, zwischen den Extremen. Sie mischen Ausdrucksstärke bei Operationen und Objekten.

In der Welt partizipativer Systementwicklung wurde die Objektorientierung oft besonders begrüßt. Der Grund ist in der Art der Anwendung wie im methodischen Vorgehen zu suchen. Partizipation trägt in sich das Ziel, die Kontrolle über die Abläufe und die Arbeitsschritte selbst bei den Akteuren zu belassen. Die Gegenstände hingegen, auf die ihre Arbeit sich erstreckt – die ominösen Dokumente aller Art – mögen sehr wohl informatisiert werden. Denn greifbar sind die Vorteile der Aktualisierung, Korrektur, Vervielfältigung, Formgebung. Lokal separierte Operationen am Objekt – die Stärke der Objektorientierung – spielen partizipativer Entwicklung in die Hände. Auf die fachlich-sachliche Diskussion dessen, was Objekt, Attribut, Wert und Klasse sein soll, kann ich mich einlassen, ohne befürchten zu müssen, dass diese Rationalisierung gleich meine ganze Arbeit schluckt.

In der partizipativ geförderten lokalen Betrachtung von Arbeitsgegenständen (Materialien) gelingt es, elementare Funktionen („Werkzeuge“) zu identifizieren, die ständig auf viele Objekte parallel anzuwenden sind. Hier treffen sich, vielleicht in einem zufälligen Augenblick soziotechnischer Entwicklung, Partizipation und Objektorientierung. Aus dem Erfolg heraus die generell gegebene Situiertheit von Software zu leugnen, kann nur zum Scheitern führen.

5 Objekte, so einfach?

Es wird manchen wundern, dass das Objekt bei den Informatikern eine prominente Rolle zu einer Zeit erlangte, als die Philosophie es gerade abschaffte. Der Versuch, es zu fassen, führte das postmoderne Denken zu seiner Auflösung. Eine tumbe Nachlässigkeit der Informatik?

Die Gegenstände der Welt, ob Ding oder Prozess, sind ihrer Stofflichkeit zu berauben, wenn sie der Software verfallen. Als ausgedehnte Materie geht nichts durch das Nadelöhr der Eingabekanäle in den Computer ein. Der einzige Weg hierzu ist die Verwandlung der Dinge in Zeichen. Das Ding draußen bleibt, was es ist, unabhängig von der Meinung der Philosophen. Sein Zeichen aber kann drinnen *Movens* beliebiger algorithmischer Prozesse werden.

Was die Informatiker und Software-Praktiker *Objekt* nennen, ist Objekt nur im abstraktesten Sinne. Jede Dinglichkeit hat es verloren. Es entspricht dem, was es vertritt, nur insoweit, als die in die Bezeichnung eingehenden Attribute und ihre Werte reichen. Wie jeder maschinelle ist auch dieser Zugang zur Welt ein reduzierender, standardisierender, filternder. Wie jeder maschinelle Zugang öffnet aber auch dieser, erst einmal gewonnen, Bewegung, die vorher nicht zu denken war.

6 Das algorithmische Zeichen

Drinnen im Computer wird alles zu *Datum*, ob es nun Programm oder Daten herkömmlicher Art war. Von den abstrakten, unsichtbaren Daten, ihren Eigenschaften und Veränderungen im Inneren der Maschine erhalten wir Kunde erst wieder, wenn sie in stoffliche, wahrnehmbare Form rückverwandelt werden und so an der Peripherie der Maschine erscheinen. Die sichtbare Form als Bild oder Text herrscht dabei vor, gelegentlich gibt es auch Klang.

Diese erscheinenden Formen von Software treffen als Signale auf unsere Sinne. Indem wir sie *wahr*nehmen und folglich interpretieren, machen wir die Signale zu Zeichen. Denn ohne die erscheinenden Signale als Entitäten[16] zu nehmen, die für anderes stehen, ohne ihnen

[16] Ein Philosoph könnte sagen: als *Etwase*.

eine Einbettung in unsere Situation zu geben, können wir sie nicht wahrnehmen. Wir interpretieren permanent und schaffen so ständig Zeichen.

Diese besitzen in unserem Fall die Merkwürdigkeit, dass sie nicht nur draußen von uns, sondern auch drinnen, vom Computer, interpretiert werden. Seine Interpretation aber ist treffender als eine *Determination* zu verstehen. Der Computer *bestimmt*, was er aus dem einlaufenden Signal machen soll. Das Signal führt parallel zur offenen Interpretation durch uns und zur geschlossenen Determination durch den Prozessor. Dem Signal wachsen zwei Interpretanten zu. Den externen (human-bedingten) nennen wir den *intentionalen Interpretanten*, den internen (computer-bedingten) den *kausalen Interpretanten* (oder Determinanten).

Wir nennen das semiotische Gebilde, das hier entsteht, *algorithmisches Zeichen*. Es ist nicht allein Wahrnehmungsangebot, sondern auch Anlass zur Kalkulation. Es kann in Bewegung gesetzt, d.h. ausgeführt werden.[17] In ihm gewinnen wir den adäquaten Begriff für Software. Software zu gestalten heißt, komplexe algorithmische Zeichen und deren Prozesse zu gestalten. Hier erscheint die Nähe zum Design zwingend. Sie bleibt nicht zufällige Laune.

Wir sehen insgesamt, dass der flüchtige semiotische Charakter von Software ihre Entwicklung dem konsekutiven, logik-orientierten Zugriff des Ingenieurs teilweise entwindet und dem alternativen, ästhetik-orientierten des Designers zuführt. Vermutlich wird die Zukunft in der Begegnung beider Kapazitäten und Haltungen zu suchen sein[18].

7 Schluss

Software ist komplexe Funktion, komplexe Benutzung, komplexe Entwicklung. Ihr semiotischer Charakter hat sie aus der Welt der stofflich-energetisch geprägten Artefakte hervortreten, ja: herausbrechen lassen. Die Methoden ihrer Entwicklung sind deswegen immer wieder umgestülpt worden. Dies wird oft zur Verkündung eines neuen Paradigma überhöht, in deutlicher Verkennung dessen, was ein Paradigma ausmacht.[19]

Eines der neuen Prinzipien tritt mit erstaunlich humanem Credo auf: das eXtreme Programming.[Lippert et al. 02] In ihm schlingt sich eine Form partizipativer Systementwicklung in die Programmiermethodik hinein. Objektorientierung erscheint als selbstverständlich, wenn gebraucht. Kooperatives Handeln im Team und nach außen prägt das Vorgehen.

Es könnte sein, dass sich eine Abkehr von dogmatisierenden Verkrustungen abzeichnet, dass Glaubenssätze über die neueste und schon wieder beste Methode ingenieurartiger Software-Entwicklung durch Haltungen abgelöst werden, die geeignet sein könnten, ihrer eigenen Verhärtung zu widerstehen. Stattdessen könnten sie heiterer Gelassenheit soviel Raum geben, dass die Komplexität der Aufgabe als bewältigbar erkannt würde. Bewältigung würde im günstigen Fall nicht dadurch erreicht werden, dass tatsächlich die immer wieder erhofften, alles umfassenden Systeme mit ökonomisch vertretbarem Aufwand auf dem Markt erschienen. Vielmehr dadurch, dass die als notwendig behauptete Komplexität von Softwaresystemen als der Popanz erkannt würde, zu dem sie regelmäßig aufgeblasen wird. Die Haltung und Herangehensweise des eXtreme Programming scheint, soweit sie Haltung bleibt und nicht zur Methode stilisiert wird, solches Potential in sich zu tragen.

So genannte sichere Systeme entwickeln zu wollen, dürfte mit Sicherheit[20] der falsche Weg sein. Unter den lebendigen Menschen hingegen die Einsicht zu stärken, dass es darauf ankomme, mit dem *Bruch* umgehen zu lernen, der bei rhizomartig durch die Infrastrukturen der Gesellschaft wuchernden technischen Systemen notwendigerweise auftritt, verspricht mehr. Die Fragilität der technischen Existenz, von der Max Bense sprach, hat uns in der Software, also der semiotischen Dimension, mit solcher Heftigkeit erreicht, dass ihr mit Vorgehensweisen der fortschrittsgläubigen traditionellen Natur- und Technikwissenschaften nicht mehr begegnet werden kann. In der Hinwendung zur Dialektik von handelndem Subjekt und behandeltem Objekt treffen sie auf Erfahrungen der hermeneutischen und gestaltenden Disziplinen. Sollte uns dies nicht Anlass zu heiterer Gelassenheit in eher trübseliger Zeit geben?

---

[17] Die Verhältnisse sind in [Nake 01] ausführlicher dargestellt.
[18] In einer Schrift arbeiten Peter Bøgh Andersen und der Autor dies genauer heraus.
[19] So jedenfalls, wenn man Thomas Kuhn in der Verwendung des Begriffes Paradigma folgt [Kuhn 73].
[20] *Pun intended*.

Literatur

[Brooks 75] Frederick P. Brooks Jr.: *The mythical man-month. Essays on software engineering*. Reading, MA: Addison-Wesley 1975

[Bolter 91] Jay David Bolter: *Writing space. The computer, hypertext, and the history of writing*. Hillsdale, NJ: Lawrence Erlbaum 1991

[Coy et al. 92] W. Coy, F. Nake, J.-M. Pflüger, A. Rolf, J. Seetzen, D. Siefkes, R. Stransfeld (Hrsg.): *Sichtweisen der Informatik*. Braunschweig: Vieweg 1992

[Dasgupta 91] Subrata Dasgupta: *Design Theory and Computer Science*. Cambridge: Cambrigde University Press 1991

[DeMarco & Lister 91] Tom DeMarco, Timothy Lister: *Wien wartet auf Dich! Der Faktor Mensch im DV-Management*. München, Wien: Hanser 1991 (Amer. Original 1987)

[Desel 01] Jörg Desel (Hrsg.): *Das ist Informatik*. Berlin u.a.: Springer Verlag 2001

[Ehn & Malmborg 98] Pelle Ehn, Lone Malmborg: The design challenge. *Scand. J. of Inf. Systems* 10, 1&2 (1998) 211-218

[Floyd et al. 92] Christiane Floyd, Heinz Züllighoven, Reinhard Budde, Reinhard Keil-Slawik (eds.): *Software development and reality construction*. Berlin u.a.: Springer Verlag 1992

[Floyd 94] Christiane Floyd: Software-Engineering – und dann? *Informatik Spektrum* 17 (1994) 29-37

[Greenbaum & Kyng 91] Joan Greenbaum, Morten Kyng (eds.): *Design at work*. Hillsdale, NJ: Lawrence Erlbaum 1991

[Kuhn 73] Thomas Kuhn: *Die Struktur wissenschaftlicher Revolutionen*. Frankfurt: Suhrkamp 1973

[Lippert et al. 02] Martin Lippert, Stefan Rook, Henning Wolf: *Software entwickeln mit eXtreme Programming*. Heidelberg: dpunkt-Verlag 2002

[Nake 92] Frieder Nake: Informatik und die Maschinisierung von Kopfarbeit. In [Coy et al. 92: 181-201]

[Nake 01] Frieder Nake: Das algorithmische Zeichen. In: W. Bauknecht, W. Brauer, Th. Mück (eds.): *Informatik 2001. Tagungsband der GI/OCG Jahrestagung 2001*. Bd. II, 736-742

[Newman & Lamming 95] William Newman, Michael Lamming: *Interactive system design*. Reading, MA: Addison-Wesley 1995

[Peirce 93] Charles S. Peirce: *Phänomen und Logik der Zeichen*. Frankfurt: Suhrkamp 1993 (2. Aufl.)

[Schefe et al. 93] P. Schefe, H. Hastedt, Y. Dittrich, G. Keil (Hrsg.): *Informatik und Philosophie*. Mannheim: BI Wissenschaftsverlag 1993

[Whitaker et al. 91] Randall Whitaker, Ulf Essler, Olov Östberg: Participatory business modeling. Research Report TULEA 1991:31. Luleå University, Schweden, 1991

[Winograd 96] Terry Winograd: *Bringing design to software*. Reading, MA: Addison-Wesley 1996

[Züllighovern 98] Heinz Züllighoven: *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz*. Heidelberg: dpunkt Verlag 1998

Ebenen der Abstraktion und ihre Abbildung auf konzeptionelle Modelle - oder: Anmerkungen zur Semantik von Spezialisierungs- und Instanzierungsbeziehungen

Ulrich Frank
Institut für Wirtschaftsinformatik
Universität Koblenz

1 Einleitung

Die Analyse realweltlicher Domänen lässt mitunter eine Reihe verschiedener Abstraktionsmöglichkeiten erkennen. Deren Abbildung mit Hilfe gängiger Sprachen der konzeptionellen Modellierung führt teilweise zu erheblichen Problemen, die kaum thematisiert werden. So kann die scheinbar offenkundige Bedeutung von Spezialisierungsbeziehungen zu kontra-intuitiven Konsequenzen führen, die die Qualität von Informationssystemen nachhaltig gefährden. Die Ursache solcher oft ungewollten Modellierungsanomalien liegt darin, dass die nach verbreiteter Meinung so natürlichen Begriffe 'Klasse' und 'Objekt' in Programmier- wie auch in Modellierungssprachen in einer Bedeutung verwendet werden, die in subtiler Form von deren Verwendung in der Umgangssprache wie auch in der Mathematik abweicht. Daneben empfiehlt die Analyse realweltlicher Phänomene mitunter eine Differenzierung von mehr Abstraktionsebenen als durch gängige Modellierungssprachen darstellbar. Dies führt zu der unangenehmen Konsequenz, dass eine als angemessen erachtete konzeptionelle Beschreibung nicht semantisch entsprechend in einem konzeptionellen Modell umgesetzt werden kann – ebenfalls mit unerfreulichen Konsequenzen für die Qualität des zu erstellenden Informationssystems. In einigen Anwendungsbereichen der (Meta-) Modellierung gibt es gleichzeitig Bedarf an Instanzierungs- und an Spezialisierungsbeziehungen. Sie können jedoch nicht gemeinsam verwendet werden, was zu erheblichen Herausforderungen führt.

Im folgenden Beitrag werden die skizzierten Probleme zunächst dargestellt und analysiert. Die Betrachtung zeigt, dass der Stand der Kunst bei Modellierungs- und Implementierungssprachen noch nicht befriedigend ist. Während dieser Umstand zu weiterer Forschung in diesen Bereichen rät, bleibt zu klären, wie man den dargestellten Probleme mit heute verfügbaren Sprachen begegnen kann. Zu diesem Zweck werden pragmatische Lösungsansätze vorgestellt und diskutiert.

2 Kontra-intuitive Semantik von Spezialisierungsbeziehungen

Generalisierung ist ein zentrales Abstraktionskonzept des menschlichen Denkens: Wir abstrahieren von der Varianz der Spezialfälle und konzentrieren uns auf einen gemeinsamen, für bestimmte Betrachtungen als wesentlich erachteten Kern. Die Umkehrung der Generalisierung, die Spezialisierung, ist ebenfalls ein gängiges Konzept zur Strukturierung unserer Weltsicht: Immer dann, wenn uns ein genereller Begriff für eine spezielle Betrachtung – etwa um verschiedenartige realweltliche Objektarten zu unterscheiden - nicht hinreicht, erlaubt uns die Einführung einer Spezialisierung eine differenziertere Beschreibung. Dabei entsteht ein spezialisierter Begriff durch das Hinzufügen weiterer Eigenschaften. Die durch den Oberbegriff festgelegten Eigenschaften gelten also weiterhin im Unterbegriff – sie werden „geerbt". Im menschlichen Denken wird diese Regel allerdings nicht immer konsequent angewendet: Mitunter werden Generalisierungen gebildet, obwohl man weiß, dass nicht für alle Spezialisierungen alle Eigenschaften der Generalisierung gelten. Ein bekanntes Beispiel dafür ist die Aussage 'Vögel können fliegen'. Wenn man einer Vogelart begegnet, für die diese Aussage nicht zutrifft, gerät das Weltbild nicht ins Wanken. Anders ist dies bei entsprechenden Formalisierungen. Sie führen zu Widersprüchen. Im Zusammenhang mit objektorientierten Programmiersprachen ist hier an die Redefinition geerbter Eigenschaften zu denken, die das berüchtigte Ko- bzw. Kontravarianzproblem ([Meye97], S. 621 ff.) nach sich ziehen. In der Künstliche Intelligenz Forschung gab es einige Ansätze, durch mehrwertige Logiken ein „non-monotonic reasoning" formal zu rekonstruieren. Wir werden diese Problematik im Folgenden allerdings ausklammern und allein solche Generalisierungen betrachten, für die keine Ausnahmen zu berücksichtigen sind.

2.1 Ein Beispiel

Generalisierung und Spezialisierung sind Konzepte, die für unseren Umgang mit einer komplexen Welt unerlässlich sind. Dementsprechend intuitiv scheint die Bedeutung von Generalisierungs- bzw. Spezialisierungsbeziehungen in der objektorientierten Modellierung. Betrachten wir dazu folgendes Beispiel: Um die Implementierung eines Informationssystems für eine Universität vorzubereiten, soll ein Objektmodell erstellt werden, in dem u. a. Studenten, wissenschaftliche Mitarbeiter und Professoren zu repräsentieren sind. Es liegt auf der Hand, dass es sich dabei jeweils um Personen handelt. Es bietet sich also an, die Klasse 'Person' als Generalisierung der Klassen 'Student', 'Assistent' und 'Professor' einzuführen. Die weitere Analyse der Domäne ergibt, dass auch Programmierer, Dozenten und Administratoren abzubilden sind. Auch dabei handelt es sich offensichtlich um Spezialisierungen von 'Person'. Es ist allerdings möglich, dass ein Programmierer gleichzeitig Student oder Assistent ist. Einfache Spezialisierungsbeziehungen erlauben es nicht, solche Zusammenhänge abzubilden. Demgegenüber scheint mehrfache Spezialisierung bzw. Mehrfachvererbung ein angemessener Ansatz zu sein, um den dargestellten Sachverhalt abzubilden. Abb. 1 zeigt ein Klassendiagramm, in dem entsprechende Spezialisierungsbeziehungen verwendet werden.

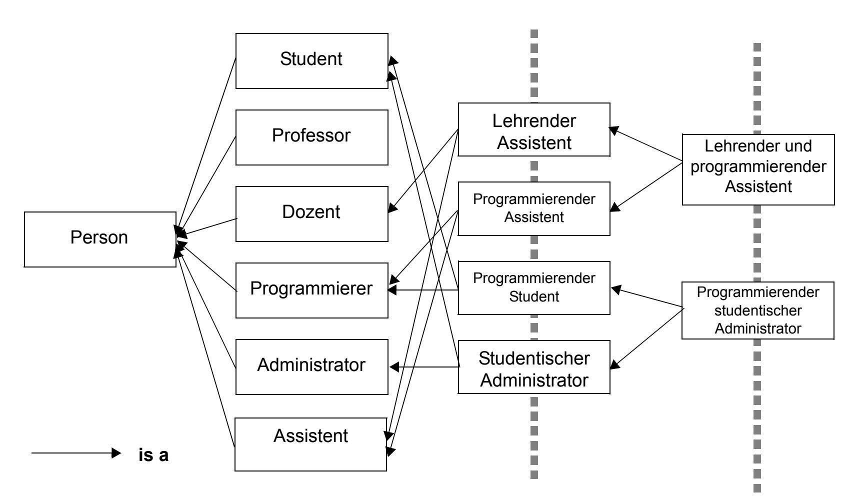


Abb. 1: Klassendiagramm mit mehrfacher Spezialisierung

Bei näherer Betrachtung der spezialisierten Klassen fällt zunächst auf, dass die Klassenbezeichner nicht 'natürlich' sind, denn sie entsprechen offensichtlich nicht gängigen Begriffen in der betrachteten Domäne. Wir reden nicht von einem 'Programmierenden

studentischen Administrator’. Nun könnte man diesen Umstand allein für einen Ausdruck eingeschränkter Verständlichkeit des Klassendiagramms halten. Tatsächlich handelt es sich hier aber um eine schwerwiegende Anomalie: Die Bedeutung des Modells weicht erheblich von der indentierten, naheliegenden Bedeutung ab. Dies gilt jedenfalls dann, wenn man - wie dies häufig in der Software-Entwicklung geschieht - die Semantik von Spezialisierungsbeziehungen in objektorientierten Programmiersprachen zugrunde legt. Auch wenn objektorientierte Modellierungssprachen zumeist keine präzise Semantik für Spezialisierungsbeziehungen festlegen, wird bei der Interpretation bzw. Transformation von Klassendiagrammen zumeist die Semantik verwendet, die in Programmiersprachen üblich ist. Das drückt sich etwa darin aus, dass die Klassen des Entwurfsmodells üblicherweise in korrespondierende Klassen der eingesetzten Programmiersprache transformiert werden. Gängige UML-Werkzeuge verfügen über Code-Generatoren, die eine entsprechende Umsetzung des Klassendiagramms in Code vorsehen [KiFr02].

Die erwähnte Anomalie ergibt sich dadurch, dass Veränderungen im Lebenszyklus der in einem solchen Modell repräsentierten Objekte mit umständlichen, wenig natürlich erscheinenden Wartungsoperationen verbunden sind, die zudem die Integrität eines Informationssystems gefährden. Betrachten wir dazu ein Objekt der Klasse ’Student’. Sobald dieser Student hinreichende Programmierkenntnisse erworben hat, kann er der Klasse ’Programmierender Student’ zugeordnet werden. In unserer alltagsweltlichen Sicht der Dinge ist eine solche Änderung wenig spektakulär: Der betrachtete Mensch bleibt nach wie vor Student und ist ergänzend dazu auch ein Programmierer. In der objektorientierten Implementierung des Modells gestaltet sich diese Änderung sehr viel aufwendiger: Zunächst ist eine neue Instanz der Klasse ’Programmierender Student’ anzulegen. Dann ist der Zustand der korrespondierenden Instanz der Klasse ’Student’ auf diese Instanz zu übertragen - einschließlich aller Referenzen auf diese Instanz! Schließlich ist die Instanz der Klasse ’Student’ aus dem System zu entfernen. Es liegt auf der Hand, dass eine solche Operation eine ernsthafte Gefährdung der Integrität eines Informationssystems darstellt.

In der Literatur zur objektorientierten Software-Entwicklung findet sich eine Reihe von Ansätzen, die diesem Problem zu begegnen versuchen. Arbeiten zur “dynamischen Klassifizierung” [MaOd98] oder “Klassenmigration” [Wier96] sind darauf gerichtet, die Konsequenzen des Problems handhabbar zu machen. Sie setzen also an den Symptomen an, nicht an der Ursache. Die Ursache für diese unterschiedlichen Interpretationen von Spezialisierungsbeziehungen liegt in den jeweils verschiedenen Klassenbegriffen. Während wir in der Alltagswelt implizit von einem extensionalen Klassenbegriff ausgehen, wird in objektorientierten Programmiersprachen zumeist ein intensionaler Klassenbegriff verwendet. Eine Klasse wird intensional durch Eigenschaften (Attribute, Operationen) definiert. Objekte werden gleichsam aus dieser Schablone instanziert. In extensionaler Wendung wird eine Klasse als eine Menge gleichartiger Objekte definiert. Der wesentliche Unterschied zwischen beiden Bedeutungen liegt in der Zuordnung von Objekten zu Klassen. Der intensionale Klassenbegriff impliziert, dass ein Objekt jeweils Instanz einer und nur einer Klasse ist. Der extensionale Klassenbegriff erlaubt es, dass ein Objekt gleichzeitig mehreren Klassen angehören kann. In diesem Fall entspricht Spezialisierung der *Subordination* [Wolt96], d. h. eine spezialisierte Klasse ist eine echte Teilmenge einer oder mehrerer Oberklassen. Ein Klasse, die aus mehreren Oberklassen spezialisiert wird, ist die Schnittmenge der Oberklassen. Wendet man den extensionalen Klassenbegriff auf unser Beispiel an, reicht es hin, sich auf die Klassen zu beschränken, die unmittelbar von der Klasse ’Person’ spezialisiert werden. Es ist nicht zwingend nötig, weitere Subklassen zu bilden, da ein Objekt gleichzeitig mehreren Klassen angehören kann. Die mengenorientierte Darstellung in Abb. 2 verdeutlicht diesen Umstand. In Datenbankschemata wird i. d. R. ein extensionaler Klassenbegriff verwendet - was allerdings in dem hier betrachteten Zusammenhang nur dann von Bedeutung ist, wenn es möglich ist, Spezialisierungsbeziehungen auszudrücken. Wenn ein Student Programmierkenntnisse erwirbt, wird das entsprechende Objekt nicht gelöscht, sondern lediglich auch der Klasse ’Programmierer’ zugeordnet (*Subsumption* [Wolt96]).

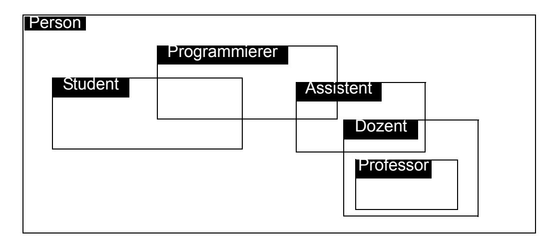


Abb. 2: Klassen mit extensionaler Semantik

Vor dem Hintergrund dieser Überlegungen scheint es ratsam, bei der Modellierung die natürlicher erscheinende extensionale Semantik von ’Klasse’ zu verwenden. Tatsächlich wird in der aktuellen Version 1.4 der UML wird offen gelassen, ob eine Klasse intensional oder extensional zu interpretieren ist. Ein Objekt kann mehreren Klassen zugeordnet werden[1]. Leider ist ein solches Vorgehen mit einem gravierenden Nachteil verbunden. Da die Implementierung von Klassendiagrammen i. d. R. mit objektorientierten Programmiersprachen erfolgt, würde die Verwendung eines extensionalen Klassenbegriffs in der Modellierung zu einem semantischen Bruch führen, der der gewünschten Integration von konzeptionellen Modellen und Code entgegenliefe. Dieser Umstand legt eine kritische Revision objektorientierter Programmiersprachen nahe. Da aber in absehbarer Zeit nicht mit einer Ersetzung der gegenwärtig vorherrschenden Sprachen zu rechnen ist, ist zunächst zu fragen, wie durch geeignete Modellierungskonzepte ein semantischer Bruch zwischen konzeptionellen Modellen und Programmen vermieden werden kann.

2.2 Ein Lösungsansatz

Um zu verhindern, dass Spezialisierungsbeziehungen in Klassendiagrammen so verwendet werden, dass sie der eigentlichen Intention des Modellierers nicht entsprechen, ist zunächst Aufklärung nötig: Modellierer müssen sich der möglicherweise kontra-intuitiven Semantik von Spezialisierungsbeziehungen bewusst sein. Ein Blick in gängige Lehrbücher zur objektorientierten Modellierung zeigt, dass hier Nachholbedarf besteht, da dieses Problem i. d. R. ignoriert wird. Dabei ist die Semantik von Spezialisierungsbeziehungen in objektorientierten Systemen ein ausführlich erforschter Gegenstand. Einige Autoren warnen ausdrücklich vor der vorschnellen Verwendung von Spezialisierung auf der Basis eines intensionalen Klassenbegriffs ([Szyp98], [Meye97]). Gleichzeitig ist in sog. klassenlosen Programmiersprachen ein von der Spezialisierung bzw. Vererbung abweichendes Konzept zur Förderung der Wiederverwendung vorgesehen. Ein Objekt, das die Eigenschaften eines anderen Objekts erben soll, wird so mit diesem assoziiert, dass eingehende Methodenaufrufe, die nicht im Protokoll des erbenden Objekts enthalten sind, transparent an das vererbende Objekt weitergeleitet werden. Dabei ist allerdings zu berücksichtigen, dass es sich hierbei nicht um Spezialisierung von Klassen handelt. Eine differenzierte Betrachtung von klassenlosen Programmiersprachen findet sich in [Male95].

1.Gleichzeitig wird allerdings die Semantik von Spezialisierungsbeziehungen nicht spezifiziert.

Ein pragmatischer Ansatz, um Spezialisierungsbeziehungen mit gängigen Modellierungssprachen sinnvoll auszudrücken und gleichzeitig den semantischen Bruch zwischen konzeptionellen Modellen und Code zu vermeiden, ist die Einführung einer ergänzenden Delegationsbeziehung. Dazu wird auf das Konzept einer 'Rolle' zurückgegriffen. Eine Klasse wird über eine Delegationsbeziehung als Rolle einer anderen Klasse, des Rolleninhabers, definiert. Dadurch erbt sie nicht die Eigenschaften der Rolleninhaber-Klasse. Vielmehr sind ihre Objekte Rollen korrespondierender Rolleninhaber-Objekte. Wenn die Rollenobjekte eine Nachricht erhalten, die in ihrem Protokoll nicht enthalten ist, wird diese transparent an den Rolleninhaber weitergeleitet - ähnlich wie bei klassenlosen Programmiersprachen. Auf diese Weise wird also nicht nur das Protokoll des Rolleninhaber-Objekts 'vererbt', sondern auch sein Zustand: Die Nachricht 'nachName', die an ein Objekt der Klasse 'Student' gesendet wird, wird an ein assoziiertes Objekt der Klasse 'Person' weitergeleitet und der dort abgelegte Name wird zurückgereicht. Abb. 3 zeigt ein Modell unserer Beispieldomäne, in dem Delegation verwendet wurde.

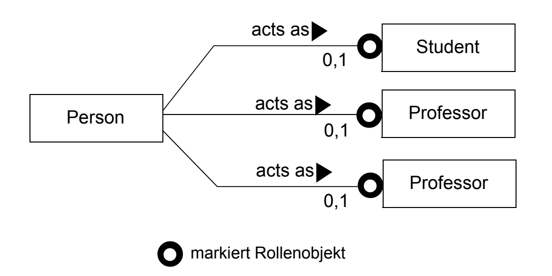


Abb. 3: Beispielmodell unter Verwendung von Delegationsbeziehungen

Delegation ist seit langem ein wichtiges Konzept in verschiedenen Feldern der Informatik - vor allem in der KI-Forschung und im Bereich der Programmiersprachen. Auch in der konzeptionellen Modellierung wurde bereits vor langer Zeit der Bedarf an einem solchen Konzept betont [BaDa77]. In der Literatur zur objektorientierten Software-Entwicklung findet sich ebenfalls eine beachtliche Zahl von - teilweise impliziten - Hinweisen auf Delegation (z. B. [BaDo96], [GoRu95], [KaSc96], [Lie86], [Rum93]). Dessen ungeachtet bietet keine der bekannten objektorientierten Modellierungssprachen, einschließlich der UML, ein entsprechendes Konzept. Eine Spezifikation der Semantik von Delegation als Modellierungskonzept findet sich in [Fran00]. Eine angemessene Verwendung dieses Konzepts empfiehlt die Berücksichtigung einiger Richtlinien:

Grundsätzlich gilt, dass man sich nicht durch den überladenen Bezeichner 'is a' verwirren lassen sollte.

- Falls eine Beziehung zwischen zwei Klassen sinnvoll 'repräsentiert' oder 'fungiert als' genannt werden kann, handelt es sich um einen Kandidaten für Delegation.
- Falls eine Generalisierung/Spezialisierung nicht denknotwendig für die gesamte Lebenszeit eines Systems gelten muss, kann Delegation eine bessere Option sein. Beispiel: Ein Professor muss nicht notwendigerweise ein Angestellter sein.
- Es gibt typische Kandidaten für Rolleninhaber-Klassen: Personen, Organisationen, Maschinen.

Die Einführung eines zusätzlichen Konzepts wie Delegation hilft dem versierten Modellierer, den Unzulänglichkeiten einer Spezialisierungssemantik entgegenzuwirken, die aus einem intensionalen Klassenbegriff resultieren. Gleichzeitig kann auf diese Weise ein semantischer Bruch zur Implementierungssprache weitgehend vermieden werden. Dennoch bleibt ein solcher Ansatz unbefriedigend, denn er beseitigt nicht die Problemursache, die in einem Klassenkonzept liegt, das vom bewährten alltagsweltlichen - und auch in den Formalwissenschaften verbreiteten - Klassenbegriff abweicht.

3 Unzureichende Abstraktionsebenen

In der Software-Entwicklung werden i. d. R. drei grundlegende Abstraktionsebenen unterschieden. Die Meta-Ebene legt die Sprache(n) zur Spezifikation der darunter liegenden Schema- oder Typ-Ebene fest. Ein solches Schema, z. B. ein Klassendiagramm, kann dann als Instanz eines korrespondierenden Metamodells angesehen werden. Die Instanzenebene schließlich dient der Beschreibung von Objekten, die aus einem Schema instanziert wurden. In der konzeptionellen Modellierung kann i. d. R. nur eine Ebene, nämlich die Schema-Ebene, manipuliert werden. Die Instanzebene ergibt sich daraus implizit. Ergänzend zu diesen grundlegenden Abstraktionsebenen kann das Abstraktionsniveau von Teilen eines Modells durch Konzepte wie Verkapselung, Polymorphie oder Generalisierung/Spezialisierung variiert werden. Die auf diese Weise verfügbaren Abstraktionsmöglichkeiten reichen allerdings häufig nicht aus, um realweltliche Sachverhalte angemessen zu beschreiben. Betrachten wir dazu die Modellierung von Ressourcen, also etwa von peripheren Geräten. In Abb. 4 sind verschiedene Abstraktionsebenen dargestellt, die wir im täglichen Umgang mit Ressourcen ohne nennenswerte Probleme differenzieren können.

Die Analyse der in Abb. 4 dargestellten Abstraktionsebenen macht zweierlei deutlich. So ist es einerseits leicht vorstellbar noch weitere Ebenen einzuführen. Beispielsweise könnten weitere Druckerklassen wie z. B. „Seitendrucker“ eingeführt werden. Im Hinblick auf die Verwendung entsprechender Konzepte zeigt sich andererseits, dass die Unterscheidung zwischen Spezialisierung und Instanzierung nicht immer intuitiv ist: Ist ein Laserdrucker eine Instanz oder eine Spezialisierung der Klasse 'Drucker'? Initialisierte Typen müssen in gängigen Systemarchitekturen als Instanzen abgebildet werden. Damit macht offensichtlich auch die “Spezialisierung” von Instanzen Sinn, wie die “Clone” genannte Erweiterung eines initialisierten Typs (also letztlich einer Instanz) zeigt. Nun könnte man dieser Kritik entgegenhalten, dass etwa die Beschreibung initialisierter Typen i. d. R. nicht Gegenstand der konzeptionellen Modellierung ist. Vielmehr soll ein konzeptionelles Modell ja bewusst von den Teilen des abgebildeten Realitätsbilds abstrahieren, die absehbaren Änderungen unterliegen. Im Einzelfall mag es allerdings wichtig sein, konkrete Ressourcentypen zu erfassen (etwa einen bestimmten Rechnertyp), weil er spezifische Implikationen für die Gestaltung von Informationssystemen mit sich bringt, die nicht vernachlässigt werden dürfen. Um eine differenzierte Modellierung zu unterstützen, sollte eine Sprache zur konzeptionellen Modellierung in jedem Fall erlauben, das Abstraktionsniveau der angebotenen Konstrukte explizit zu machen.

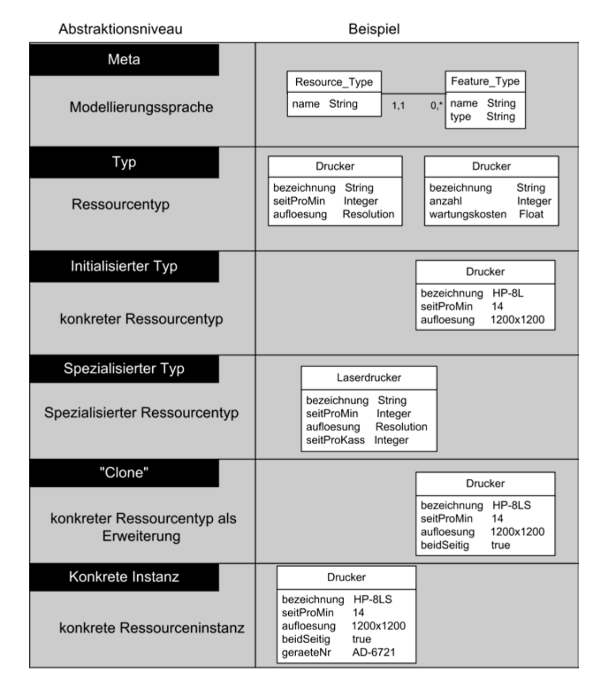


Abb. 4: Abstraktionsebenen bei der Modellierung von Ressourcen

Gängige Modellierungssprachen - wie auch Implementierungssprachen - bieten hier offensichtlich keine angemessene Unterstützung. Die einzige Möglichkeit, die dem Modellierer bzw. Software-Entwickler bleibt, ist die Überladung von Abstraktionsebenen. So kann etwa eine entsprechende Anwendung einzelne Klassen in einem Datenbankschema als Metaklassen interpretieren. Ein Beispiel dafür findet sich in [Fran02]. Ein solcher Ansatz bleibt allerdings unbefriedigend, weil er einerseits zu kaum verständlichen Modellen führt und andererseits erfordert, dass die jeweils vorgesehenen Interpretationen implementiert werden müssen.

4 Spezialisierung versus Instanzierung

Die Unterscheidung zwischen Spezialisierung und Instanzierung erscheint unproblematisch. Sieht man von der Mehrdeutigkeit des in beiden Fällen gern benutzten Bezeichners 'is a' ab, sind die Verwendungskontexte beider Konzepte i. d. R. deutlich unterschiedlich. Es gibt allerdings Fälle, in denen beide Modellierungskonzepte angemessen erscheinen, eine gemeinsame Verwendung sich allerdings ausschließt. So zielt die Metamodellierung darauf, für einen bestimmten Verwendungskontext, etwa die Beschreibung von Geschäftsprozessen, die Menge zulässiger Modelle zu definieren oder, anders gewendet, eine Modellierungssprache zu spezifizieren. Die Instanzen der (Meta-) Typen des Metamodells sind dann Typen, die zur Modellierung auf der Objektebene verwendet werden. Im Hinblick auf die Unterstützung des Modellierers wäre es wünschenswert, die Gemeinsamkeiten, die für alle Prozesstypen, aber auch für alle Prozessinstanzen gelten, zu erfassen. Auf diese Weise wird der Modellierer davon entlastet, im Einzelfall Konzepte zu (re-) konstruieren, die allen Modellierungsgegenständen gemeinsam sind. In einem Metamodell werden allerdings nur die Gemeinsamkeiten von Typen berücksichtigt. So sind aggregierte Prozesstypen aus anderen Prozesstypen zusammengesetzt. Es können allen Prozesstypen Attribute wie 'Anzahl von Instanzen', 'durchschnittliche Laufzeit' etc. zugeordnet werden. Diese gemeinsamen Eigenschaften von Prozesstypen werden sinnvollerweise in der Modellierungssprache spezifiziert und damit für den Modellierer wiederverwendbar gemacht. Der Ausschnitt eines entsprechenden Metamodells in Abb. 5 verdeutlicht diesen Umstand.

Die Metamodellierung hat jedoch Grenzen, da sie es nicht erlaubt, Gemeinsamkeiten von Instanzen auszudrücken. So gilt für jede Prozessinstanz, dass sie eine Anfangszeit und eine Endzeit hat. Dieses Wissen kann aber in einem Metamodell nicht untergebracht werden, da es sich bei beiden Attributen ja nicht um Eigenschaften von Prozesstypen handelt. Im Unterschied dazu würde die Verwendung generischer Typen, die durch Spezialisierung (und nicht durch Instanzierung) an individuelle Bedürfnisse angepasst werden können, die Zuweisung von Attributen, die für alle Instanzen gelten, erlauben. Hinsichtlich des Ziels, die Wiederverwendung gemeinsamer Eigenschaften zu unterstützen, konkurrieren Instanzierung und Spezialisierung also. Sie können allerdings nicht gemeinsam in einem Metamodell verwendet werden, weil die jeweils intendierten Abstraktionsebenen nicht kompatibel sind.

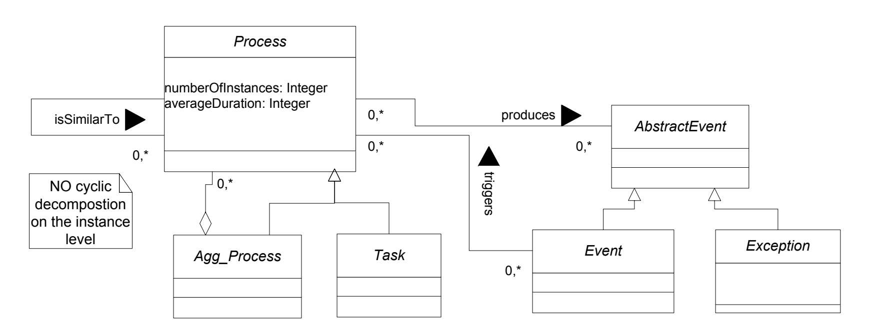


Abb. 5: Ausschnitt des Metamodells einer Prozessmodellierungssprache

Wenn man in einem Metamodell auch Eigenschaften von Instanzen ausdrücken möchte, bleibt lediglich die Einführung zusätzlicher Integritätsbedingungen für die Zustände der Instanzen der Instanzen der Typen des Metamodells. Das ist nicht eben komfortabel und führt zudem zu schwer verständlichen Modellen.

Im Unterschied zu den zuvor diskutierten Problemen handelt es sich hier allerdings nicht vordergründig um einen Mangel von Modellierungssprachen allgemein, sondern um ein Defizit solcher Sprachen, die zur Metamodellierung verwendet werden. Da dabei oft wie auch auf der Objektebene ERM-ähnliche Sprachen verwendet werden, bleibt zu vermuten, dass die Metamodellierung dedizierte Sprachen erfordert.

5 Abschließende Bemerkungen

Die Standardisierung konzeptioneller Modellierungssprachen, die seit einiger Zeit vor allem im Rahmen der UML erfolgt, verspricht eine Reihe unstrittiger Vorteile. Gleichzeitig zeigt die Betrachtung zentraler Konzepte wie Klasse, Objekt, Spezialisierung und Instanzierung, dass

gängige Modellierungssprachen wie auch die UML nicht überzeugend sind. Denn sie unterstützen einen zentralen Anspruch der konzeptionellen Modellierung nur eingeschränkt, nämlich eine natürliche, also mit vertrauten Konzeptualisierungen korrespondierende Beschreibung von Systemen. Dabei ist allerdings zu berücksichtigen, dass diese Defizite nicht allein den Modellierungssprachen angelastet werden können. Vielmehr resultieren sie zum einen aus den Eigentümlichkeiten gängiger Programmiersprachen, zum anderen aus der doppelten Zielsetzung der konzeptionellen Modellierung: möglichst natürliche Repräsentation *und* die gleichzeitige Vorbereitung anschließender, eng integrierter Implementierungsschritte.

Da die betrachteten Probleme keinesfalls als semantische Spitzfindigkeiten abgetan werden können, sondern u. U. zu einer erheblichen Beeinträchtigung der Qualität von Informationssystemen beitragen, bleibt zu fragen, wie man ihnen wirksam begegnen kann. Zunächst ist es wichtig, in der Lehre mit Nachdruck auf diese Probleme hinzuweisen und so bei den Modellierern das Bewußtsein zu schaffen, das nötig ist, um diesen Fallstricken des objektorientierten Entwurfs zu entgehen. Für die Forschung ergeben sich m. E. zwei wesentliche Konsequenzen. So sollte trotz der scheinbaren Konsolidierung, die mit der Standardisierung der UML verbunden ist, die Forschung im Bereich der Modellierungssprachen weiter geführt werden - unabhängig davon, ob die Ergebnisse von der UML abweichen oder nicht. Die zweite Konsequenz ist grundsätzlicher Art. Angesichts des Umstands, dass es sich hier nicht um ein isoliertes Problem von Modellierungssprachen handelt, scheint es mir angemessen, gegenwärtige Formen der Arbeitsteilung und Koordination in der Informatik zu überdenken. So würde eine bessere Zusammenarbeit zwischen betroffenen Arbeitsgebieten, hier ist u. a. an die Software-Technik, Datenbanken, aber auch an die Künstliche Intelligenz-Forschung zu denken, nicht nur Redundanzen vermeiden und Synergien befördern, sondern könnte auch dazu beitragen, die aufgezeigten Friktionen zwischen Modellierungssprachen und Implementierungssprachen zu vermeiden.

Literatur

[BaDa77] Bachman, C.W.; Daya, M.: The role concept in data models. In: Proceedings of the 3rd International Conference on Very Large Databases 1977, pp. 464-476

[BaDo96] Bardou, D.; Dony, C.: Split Objects: a Disciplined Use of Delegation within Objects. In: Proceedings of the OOPSLA'96. New York: ACM 1996, pp. 122-137

[Fran00] Frank, U.: Delegation: An Important Concept for the Appropriate Design of Object Models. In: Journal of Object-Oriented Programming. Vol. 13, No. 3, June 2000, pp. 13-18

[Fran02] Frank, U.: Modeling Products for Versatile E-Commerce Platforms - Essential Requirements and Generic Design Alternatives. In: Mayr, H. (Ed.): Proceedings of the 2nd International Workshop on Conceptual Modeling Approaches for E-Business (ECOMO). Berlin, Heidelberg etc.: Springer 2002

[GoRu95] Goldberg, A.; Rubin, K.S.: Succeeding with Objects. Decision Frameworks for Project Management. Reading/Mass. etc.: Addison-Wesley 1995

[KaSc96] Kappel, G.; Schrefl, M.: Objektorientierte Informationssysteme. Konzepte, Darstellungsmittel, Methoden. Wien, New York: Springer 1996

[KiFr02] Kirchner, L.; Frank, U.: Evaluierung von UML-Modellierungswerkzeugen. In: Objektspektrum, Nr. 1, 2003, S. 45-50

[Lie86] Lieberman, H.: Using prototypical objects to implement shared behavior in object-oriented systems. In: OOPSLA, 1986, pp. 214-223

[Male95] Malenfant, J.: On the Semantic Diversity of Delegation-Based Programming Languages. In: Proceedings of the OOPSLA95. New York: ACM Press 1995, pp. 215-230

[MaOd98] Martin, J.; Odell, J.: Object-Oriented Methods: A Foundation. Englewood Cliffs: Prentice-Hall 1998

[Meye97] Meyer, B.: Object Oriented Software Construction. 2. Ed., Englewood Cliffs: Prentice Hall 1997

[Rum93] Rumbaugh, J. et al.: Object Oriented Modeling and Design. Englewood Cliffs: Prentice Hall 1993

[Szyp98] Szyperski, C. A.: Component Software: Beyond Object-Oriented Programming. Reading/Mass. et al.: Addison-Wesley 1998

[Wie95] Wieringa R.J., Jonge W. de, Spruit P.A.: Using Dynamic Classes and Role Classes to Model Object Migration. In: Theory and Practice of Object Systems, 1, 1995, pp. 61-83

[Wolt96] Wolters, G.: Subordination. In: Mittelstraß, J. (Hg.): Enzyklopädie Philosophie und Wissenschaftstheorie. Mannheim: BI Wissenschaftsverlag 1996, S. 132-132

Sprachanalyse, Metadaten, *Social Navigation* – Semantik-Konzepte im Wandel

Sabrina Geißler
Universität Paderborn
Fürstenallee 11
33102 Paderborn
E-Mail: sabrina@upb.de

Kurzfassung

Die effektive Verwaltung von Wissen und die damit verbundene automatisierte Erschließung von Bedeutung stellt ein Grundproblem in der Geschichte des Computers dar. Konzentrierte sich die KI-Forschung überwiegend auf die Inhaltsanalyse von Texten, so greifen gegenwärtige Projekte im Bereich des *Semantic Web* auf manuelle Auszeichnungsverfahren zurück. Als Alternative zu diesen beiden Ansätzen möchte der vorliegende Beitrag eine dritte Sicht auf Semantik anbieten, die den Nutzer mit einbezieht und davon ausgeht, dass Bedeutung als Kondensat aus dessen Such- oder Navigationsaktivitäten entsteht. Möglichkeiten zur Beobachtung solcher Prozesse sind einerseits durch das Konzept der *Social Navigation*, andererseits durch die Suchmaschinen gegeben.

1. Einleitung

Die Verwaltung von Wissen stellt immer wieder neue Anforderungen an die Entwicklung von Technik. Dies gilt insbesondere für die elektronischen Medien, da hier der Zugriff auf ein beständig wachsendes Volumen an Dokumentenbeständen immer effizientere und effektivere Selektionsmechanismen verlangt. Technologien zum Wissensmanagement sind somit zunehmend mit der Notwendigkeit konfrontiert, „Bedeutung" beim automatisierten Erschließen eines nahezu unendlichen Textuniversums mit einzubeziehen.

Bereits sehr früh konnte die KI-Forschung mit geeigneten Lösungsansätzen zu diesem Problem aufwarten. Der Schwerpunkt lag auf dem Entwurf „intelligenter" Systeme, die den kognitiven Fähigkeiten des Menschen nachgebildet waren. Mittels inhaltsanalytisch orientierter Zugriffsverfahren und einer rein formalen Semantik wurden Texte ausgewertet und so Ambiguitäten, beispielsweise bei der Existenz von Homonymen oder syntaktischer Konstruktionen, ausgeschlossen.

Aktuell bieten Metadaten und Ontologien eine weitere Möglichkeit zur automatisierten Erschließung von Bedeutung. Als technologische Antwort auf verschiedene Defizite beim Auffinden, Extrahieren, Erzeugen oder einfach *Nutzen* von Informationen sieht das von Tim Berners-Lee konzipierte *Semantic Web* ausdrücklich Mechanismen zum Annotieren von Dokumenten vor. Die auf diese Weise erhaltenen Metadaten können dann in einem weiteren Schritt von Maschinen verarbeitet werden.

Wenn auch sowohl die inhaltsanalytische Vorgehensweise der KI-Entwicklergemeinde als auch die expliziten Kategorisierungsverfahren des *Semantic Web* sicherlich jeweils einen sehr pragmatischen Nutzen erfüllen, ist jedoch in beiden Fällen das visionäre Gesamtkonzept im Hinblick auf eine universelle Anwendbarkeit der zugrunde liegenden Semantikmodelle eher kritisch zu beurteilen. Ich möchte im Folgenden somit eine alternative Sicht auf Semantik zur Diskussion stellen, die die tatsächlichen Aktivitäten eines Benutzers mit einbezieht und davon ausgeht, dass sich Bedeutung als Kondensat aus dessen Such- oder Navigationsprozessen ergibt. Diese Vermutung ist zu präzisieren.

Aus dem eingangs formulierten Grundproblem resultiert folgende dreigeteilte Sicht auf Semantik, die auch die Chronologie ihrer Entwicklung widerspiegelt:

1) *Inhaltsanalytische Vorgehensweise der KI:*
   Entwicklung einer formalen Semantik durch Modellierung kognitiver Prozesse.
2) *Explizite Auszeichnungsverfahren durch Metasprachen (Semantic Web):*
   Verwendung eines Sets von Metadaten zur Auszeichnung von Dokumenten.
3) *Implizite Semantik durch Navigations- oder Suchprozesse:*
   Beobachtung der tatsächlichen Aktivitäten eines Benutzers zum Erstellen von Gliederungsstrukturen.

Der dritte Ansatz rückt den Zusammenhang von Struktur und Bedeutung in den Mittelpunkt – ein Aspekt, der bereits bei de Saussure[1] explizit beschrieben wurde, aber aufgrund der mitunter sehr einseitigen Orientierung der Informatik auf die traditionelle Semiotik nach Peirce häufig in Vergessenheit geraten ist.[2] Selbst eine zusätzliche Aufspaltung der Komponenten des triadischen Zeichens[3] bietet hinsichtlich meiner Untersuchung keinen Mehrwert, da sie außerstande ist den Entstehungsprozess von Bedeutung zu reflektieren.

Der von mir gewählte Untersuchungsansatz ist hypothetisch formuliert und stützt sich auf Erkenntnisse aus unterschiedlichen Argumentationszusammenhängen[4]: Er erhebt somit keineswegs den Anspruch auf Einordnung in eine bestehende Wissenschaftstradition, sondern möchte das Bewusstsein für einen neuen Problembereich schärfen: Können sich technische Artefakte angesichts der Aktivitäten des Benutzers dynamisch und flexibel anpassen und sogar verändern? Als Ausgangspunkt für meine Überlegungen möchte ich das Konzept der *Social Navigation* vorstellen, mittels dessen „reale Suchprozesse" protokolliert werden, die die Aktivitäten des Nutzers transparent erscheinen lassen. Unter Rückgriff auf zwei theoretische Modelle, einmal aus der Softwaretechnik, einmal aus der Medientheorie, wird die kulturwissenschaftliche Relevanz des angesprochenen Phänomens begründet und ein möglicher Untersuchungsansatz abgeleitet.

2. Inhaltsanalyse der KI

Die von der KI-Forschung angebotenen Semantik-Modelle gehen grundsätzlich von einem Dualismus von Mensch und Maschine aus (vgl. Haugeland 1987:75ff.). Dies suggeriert eine zweigeteilte Sicht auf Bedeutung, da die Unterscheidung zwischen einer natürlichsprachlichen und einer formalen Semantik implizit vorausgesetzt wird. Im Laufe ihrer über fünfzigjährigen Geschichte entwickelte und verfeinerte die KI ein Sprach- bzw. Semantikmodell für den Computer[5], das auf die Modellierung kognitiver Prozesse ausgerichtet war. Ausgehend von einer funktionalen Analogie zwischen einem menschlichen

[1] Vgl. hierzu die Ausführungen zum „Wertbegriff": „La valeur, prise dans son aspect conceptuel, est sans doute un élément de la signification [...]" (de Saussure 1967:158) oder auch: „Puisque la langue est un système dont tous les termes sont solidaires et où la valeur de l'un ne résulte que de la présence simultanée des autres." (de Saussure 1967:159)

[2] Vgl. hierzu z. B. die Ansätze zur Computersemiotik in Andersen (1990) sowie darauf Bezug nehmend Nake (2001).

[3] Nake schlägt hier eine Spaltung des Signifikats in eine *intentionale*, für den Menschen verständliche und eine *kausale*, für den Computer zur Ausführung von Befehlen notwendige Seite vor (Nake 2001:740]).

[4] Die vorgeschlagene dreigeteilte Sicht auf Bedeutung entspricht zwar nicht den üblichen semiotischen Gliederungskategorien, die, ganz im Sinne der strukturalistischen Tradition de Saussures, die Beziehung von Sprache und Welt untersuchen und ausgehend von einem Dualismus von Objekt und Bedeutung, die Semantik, neben Syntax und Pragmatik, in ein weiteres Feld einordnen. Dennoch bietet sich das Modell zur Untersuchung des Problemfeldes an.

[5] Einen prägnanten Überblick über die Phasen der KI-Forschung gibt z. B. Schefe (1986:30-46).

Gedächtnis und einem maschinellen Speicher wird der Computer als Medium für die Charakterisierung geistiger Prozesse und deren Formalisierung begriffen.

Der Ausdruck „Formale Semantik" bedeutet nichts anderes, als dass die Auswahlmöglichkeiten bzw. Assoziationen, die in einer natürlichen Sprache implizit gegeben sind, explizit gemacht werden, oder, wie es Dreyfus formuliert, geht es im Wesentlichen darum, „alle semantischen Überlegungen (Rückgriff auf Bedeutungen) auf die Techniken der syntaktischen (formalen) Manipulierung zu reduzieren" (Dreyfus 1984:18). Jedoch steht nicht so sehr die Frage im Mittelpunkt, ob ein Computer tatsächlich in der Lage ist eine Anweisung zu *verstehen* oder ob er nur *Verständnis simuliert*: Die Modelle nehmen eine Spaltung von Semantik vor, indem sie ein starres, regelbasiertes und von expliziten Anweisungen abhängiges Semantikmodell für den Rechner einem menschlichen, selbstregulativen und durch Assoziationen bestimmten Bedeutungsnetz gegenüberstellen.

Die Einbeziehung einer Handlungsebene sowie diverser metasprachlicher Indikatoren, die die natürlichsprachliche Kommunikation situativ prägen, erwies sich bei der Modellierung von Bedeutung zunehmend als eine Herausforderung. Sehr häufig wurde somit der Einwand formuliert, die KI müsse zwangsläufig am Kontextbegriff scheitern, je mehr Alltagsbezüge angesprochen wären. Diese Kritik wurde u. a. durch die Arbeiten von Minsky widerlegt, der so genannte situative *Frames* entwarf, die sämtliche, für eine Situation kennzeichnende Informationen beinhalten (vgl. Minsky 1994:207ff.).

Bei ihrem Entwurf von Semantik-Modellen bleibt die KI-Forschung meist in der triadischen Zeichenrelation verhaftet, da sie neben der Objekt – Inhalt-Beziehung noch eine pragmatische Dimension einführen. Die Frage nach den Grenzen der Modellierbarkeit von Kontext ist jedoch höchstens eine prinzipielle. Die KI-Kritik müsste m. E. an einem viel grundsätzlicheren Punkt ansetzen und den Entstehungsprozess sowie die Strukturgebundenheit von Bedeutung infrage stellen. Auch hierzu hat eine Forschungsrichtung der KI ein Resultat geliefert, auf das ich im Folgenden meinen Semantik-Begriff stützen möchte: Die theoretischen Modelle des Konnektionismus beschreiben Bedeutung als Funktion eines Systemzustandes – eine Auffassung, die sehr stark an die strukturalistische Wertetheorie erinnert. Das Speichern von Informationen erfolgt hier nicht mehr in genau adressierten Speicherplätzen, sondern in Netzwerken. Bedeutung präsentiert sich somit als ein relationales Gefüge, in dem jeder *Wert* von den benachbarten Werten abhängig ist. Dies ist als Fazit zunächst festzuhalten.

3. *Semantic Web* und Ontologien

Aktuelle Forschungsansätze zum *Semantic Web* bieten ebenfalls eine Reaktion auf Engpässe beim Explorieren der im Internet akkumulierten Wissensbestände. Grundlegende Idee dieser relativ jungen Webtechnologie ist es, Informationen in einer Weise aufzubereiten, dass Maschinen effektiver und effizienter auf sie zugreifen können.[6] Das „semantische Netz" enthält neben der ursprünglichen Struktur aus Dokumenten und Links noch weitere Schichten[7], die zusätzliche semantische Informationen transportieren und in einer maschinenverständlichen Form aufbereiten. Wenn auch diese Vorgehensweise zunächst altbekannte KI-Konzepte evoziert, werden die Projekte aus dem Bereich des *Semantic Web* jedoch ausdrücklich nicht, so der Erfinder Tim Berners-Lee selbst, in die Tradition der Künstlichen Intelligenz-Forschung eingereiht, da die „semantische" Auszeichnung von menschlichen Kodierern vorgenommen wird (vgl. Berners-Lee, Hendler, Lassila 2001).

[6] Einen Überblick über den gegenwärtigen Stand in der Entwicklung des *Semantic Web* geben z. B. Berners-Lee, Hendler, Lassila (2001) oder auch Geroimenko (2003:3ff.).
[7] Die Hierarchie des *Semantic Web* besteht aus mehreren Schichten, die unterschiedliche Granularitäten der formalen Definition von Bedeutung aufweisen.

Ontologien – im Schichtenmodell sind sie auf einer mittleren Ebene anzusiedeln – bieten ein Grundgerüst für das Erstellen von Metadaten. Sie werden von hochspezialisierten *Scientific Communities* genutzt, um Konzepte einheitlich zu benennen. Im Falle der Ontologien verschmelzen informatische mit philosophischen Ansätzen. Ursprünglich ein Begriff im Singular als „Lehre vom Sein" bzw. von den *Gegebenheiten* des Seins hat sich der Ausdruck in der Informatik inzwischen im Plural etabliert (vgl. Schefe, 2001:788). Dieser Begriff als Technologie zur Strukturierung von Wissen wird hier abermals von der Informatik neu definiert: Im Gegensatz zum ontologischen Anspruch der KI-Forschung, der auf eine formalisierte Darstellung des gesamten Weltwissens ausgerichtet war, verfährt man nun bescheidener und wählt Ontologien zur Kategorisierung eines sehr begrenzten, ausgewählten Bereichs des Gesamtvokabulars. Das Erfordernis einer formal präzisierten Semantik ist in diesem Zusammenhang gerechtfertigt, denn Maschinen benötigen explizite Anweisungen um Bedeutung erfassen zu können. Dennoch erscheint es mir, als werde das als „Web der nächsten Generation" oder einfach als „Zukunft des Web" (vgl. Berners-Lee, 2000:216ff.) umschriebene Konzept zu euphorisch gehandhabt.

Ich möchte hier vor allem zwei Einwände gegen eine semantische Auszeichnung mittels Metadaten formulieren, die nicht so sehr den pragmatischen Nutzen von Metadaten im Einzelfall als vielmehr das visionäre Gesamtkonzept betreffen: Im Hinblick auf den Umgang mit Quantitäten waren Metadaten im Bibliotheksbereich bzw. Archivwesen schon immer eine gängige Praxis. In diesem Zusammenhang stellen auch Bibliothekskataloge, Indices und Sigel nichts anderes als semantische Zusatzinformationen dar, die den Zugriff auf die akkumulierten Datenbestände erleichtern. Ebenso dürften Erfahrungen mit Katalogisierungssystemen in der Art von *Yahoo!* gelehrt haben, dass eine explizite Auszeichnung Mühe kostet, nie vollständig ist und somit das Semantik-Problem nicht dauerhaft lösen kann. Zweitens ist festzuhalten, dass Ontologien vor allem auf die Beschreibung eines statischen Endzustandes ausgerichtet sind, aber kaum Verlaufsstufen mit einbeziehen. Dies ist jedoch weder angesichts der Dynamik einer natürlichen Sprache gerechtfertigt, noch im Hinblick auf die relativ offenen Strukturen der Gemeinschaften, die solche Sets von Metadaten konzipieren: Entscheidungs- oder Diskussionsprozesse werden im Endergebnis nicht festgehalten.

Für den Fall, dass *Semantic Web*-Technologien den an Wissensmanagementsysteme generell gestellten Anspruch, eine „aktivere" Rolle in Erkenntnisprozessen einzunehmen (vgl. Kamphusmann 2001:744), einlösen sollen, müsste das Augenmerk weniger auf den Dualismus von Mensch und Maschine, sondern verstärkt auf Interaktion gerichtet sein. Gesucht wird ein hybrider Zugang, der es erlaubt, Nutzungsprozesse mit einzubeziehen und somit zu klären, auf welche Weise sich Strukturen im Netz herausbilden können.

4. Alternative Sicht auf Semantik: Bedeutung als Struktur

Ich möchte nun im dritten und letzten Teil dieses Artikels eine alternative Sicht auf Semantik zur Diskussion stellen, die das wechselseitige Verhältnis von Benutzung und Struktur thematisiert oder, anders formuliert, die untersucht, wie sich Strukturen im Internet auf Basis der Aktivitäten der Benutzer aufbauen. Diese Annahme bedarf der Erläuterung.

Eine natürlichsprachliche Semantik unterliegt dem permanenten Wandel und der Anpassung. Bedeutung gilt als das Resultat eines Abstraktionsprozesses, dem konkrete Sprachakte vorausgehen. Bedeutung entsteht jedoch auch immer in einem kooperativen Prozess – ein Aspekt, der aufgrund seiner offensichtlichen Trivialität sehr häufig vergessen wird: Die Sprache ist demnach überhaupt das einzige wirklich kooperative Medium, das als Ergebnis einer Übereinkunft in seiner Gesamtheit nur im intersubjektiven Raum existiert und in dem

sich Bedeutung als Konventionalisierung sozialer Praxen ergibt.[8] Im Gegensatz zu einer formalen Semantik ist eine natürliche Sprache zudem stets einem Anpassungsdruck sowie der Dynamik der Veränderung ausgesetzt. An dieser Stelle sei noch einmal auf de Saussure verwiesen, dessen historische Bedeutung bezüglich der späteren Begründung des Strukturalismus vor allem darin liegt, dass er als Erster Bedeutung als eine relationale Struktur beschrieb (vgl. de Saussure 1967:158ff.).

Das Internet selbst stellt sich nun auch als eine Struktur aus zwei Ebenen dar: Auf der einen Seite gibt es die relativ starre, durch Links miteinander verbundene Struktur der Angebote, auf der anderen Seite stehen die Nutzerbewegungen, die diesen Verweisen dynamisch folgen und höchstens durch Zugriffsstatistiken sichtbar gemacht werden können. Diese beiden Ausdrucksformen weisen gewisse Affinitäten zu den zwei „Seinsweisen" von Sprache, System und Rede, auf; sie sind, so die Vermutung Winklers, vermutlich auch auf ähnliche Weise wechselseitig aufeinander bezogen (Winkler 1997:186f.).

Zu untersuchen bleibt, auf welche Weise sich die Aktivitäten des einzelnen Benutzers, beispielsweise in Form von Navigationsprozessen oder Suchaktivitäten, auf die Gesamtstruktur „Internet" niederschlagen. Ist es also für das Internet ebenso gegeben, dass sich durch die Häufigkeit der Nutzung Angebote verdichten, wie es beispielsweise für die Einschaltquoten im Fernsehen zu beobachten ist? Kann man davon ausgehen, dass sich durch die Nutzungsaktivitäten selbst Hierarchien herausbilden, d. h. dass sich Haupt- und Nebenwege abzeichnen, die zur Herausbildung von Informationszentren führen und andere Angebote in die Peripherie verdrängen? Können Nutzerbewegungen die Struktur des Angebotes bestimmen? Ist es möglich, dass sich Bedeutung aus den Gebrauchsstrukturen selbst ergibt?[9]

Zur Vertiefung dieses Sachverhaltes möchte ich eine Untersuchung in drei Schritten anbieten: Das Konzept der *Social Navigation* (vgl. Dourish, Chalmers 1994) verdeutlicht zunächst, inwiefern die Dimension der Handlung in Algorithmen implementiert sein kann. Darauf aufbauend möchte ich zwei theoretische Ansätze heranziehen, die sich dem Sachverhalt aus unterschiedlichen Perspektiven annähern: Den skandinavischen Ansatz zur partizipatorischen Softwareentwicklung nach Christiane Floyd sowie das von Hartmut Winkler entwickelte „Modell" zur Medientheorie, das das Wechselspiel von Diskursen und Struktur auf allgemeiner Ebene untersucht.

4.1 *Social Navigation*

Für die Untersuchung konkreter „Umbauaktionen" im Netz bietet sich die Beobachtung von Nutzungsaktivitäten selbst an. Bereits 1994 entwickelten Dourish/Chalmers das Konzept der *Social Navigation* mit dem Ziel, dynamische Gliederungsstrukturen im WWW zu erzielen. Als eine von drei[10] Metaphern zur Gestaltung und Benutzung des Informationsraumes bedient sich die „soziale Navigation" des Benutzerverhaltens, um Gliederungskategorien und Beziehungen der Inhalte untereinander herzustellen.

Der Besucher einer Site generiert sowohl *aktiv* als auch *passiv* eine Struktur aus Links, Beziehungen oder Inhalten, anhand derer spätere Benutzer einen gefilterten Überblick über die Seitenansicht erhalten. Im Falle der aktiven *Social Navigation* wirkt der Benutzer eines Informationsraumes explizit auf die Struktur der Inhalte ein, dadurch dass er die relevanten

[8] Diesen Zusammenhang stellt noch einmal ausdrücklich Winkler (2002: 302ff.) heraus.
[9] Diese Fragen formulierte Winkler bereits in *Docuverse* (Winkler 1997a:338).
[10] Die Autoren unterschieden drei Arten von Metaphern für die Gestaltung und Benutzung des Informationsraumes. Neben der *Social Navigation* führen sie den Begriff der *Spatial Navigation* ein, der auf räumlichen Vorstellungen von der Welt basiert sowie die *Semantic Navigation*, bei der die Suche nach semantischen Relationen erfolgt.

Informationen nach ihrer subjektiven Beurteilung bewertet und sortiert und diese weiteren Benutzern als Orientierungshilfe zur Verfügung stellt.

Ausschlaggebend für die Herausbildung dynamischer Strukturen ist vor allem auch das Konzept der passiven *Social Navigation*, bei der das System exakt mitprotokolliert (beispielsweise durch ein entprechendes Java-Applet zur Aufzeichnung von Benutzerpfaden), in welcher Reihenfolge der Besucher einer Site diverse Einzelinformationen aufgerufen hat. Diese Informationen werden weiteren Benutzern als Navigationshilfe angeboten, beispielsweise in Form einer vom System generierten Sitemap. Durch häufige Benutzung solcher *Footprints* bilden sich virtuelle „Trampelpfade" heraus, die sich – je nach Intensität der Benutzung – vertiefen aber auch abschwächen können bzw. ganz herausfallen.[11]

Das Einbeziehen von Nutzerbewegungen stellt somit eine neue Art der Informationsgewinnung dar und bietet vor allem aber eine veränderte Sicht auf den bereits erwähnten Kooperations- bzw. Interaktionsaspekt: In einer vernetzten, arbeitsteilig organisierten Gesellschaft erhält Kommunikation einen neuen Stellenwert, da der Austausch nicht mehr vornehmlich zwischen zwei Individuen stattfindet, sondern sich der Einzelne mit dem System auseinandersetzt; es liegt somit die Vermutung nahe, dass das Internet tatsächlich Strukturen herausbilden könnte, die denen der Sprache sehr nahe kommen.

4.2 Komplementärität von Produkt und Prozess: Ein Ansatz aus der Softwaretechnik

Zurück zu der eigentlichen Ausgangsfrage: Ist das *Semantic Web* langfristig wirklich in der Lage, auf die im Internet akkumulierten Wissensbestände angemessen zu reagieren? Diesbezüglich wurde bereits auf ein Defizit verwiesen, das in der einseitigen Orientierung der Ontologien auf der Beschreibung eines fixierten Endzustandes liegt. Prozesse im Einsatzumfeld von Metadaten fallen somit systematisch aus sämtlichen Kategorisierungsversuchen heraus.

In den Achtziger Jahren des letzten Jahrhunderts entwickelte die Informatik im skandinavischen Raum ein Modell, das als Paradigmenwechsel innerhalb der Softwaretechnik gelten kann.[12] Dieser Ansatz, der die komplementäre Sicht auf Produkte und Prozesse im Entwicklungs- und Designprozess von Software in den Mittelpunkt stellt, wurde von Christiane Floyd in einem Aufsatz aus dem Jahre 1987 ausführlich beschrieben und von Reinhard Keil-Slawik in den Neunziger Jahren maßgeblich weiterentwickelt und ausgestaltet (vgl. Keil-Slawik 2000:200ff.). Im Hinblick auf meine eingangs formulierte Forschungsfrage könnte dieses Modell eine Erklärung dafür liefern, wie sich Strukturen als Ergebnis eines Abstraktionsprozesses herausbilden.

Die komplementäre Sicht auf Produkt und Prozess geht davon aus, dass sämtliche Erzeugnisse im Entwicklungsprozess von Software, z. B. Programmcode, Begleittexte oder das fertige System selber, immer in unmittelbarem Zusammenhang mit ihrer Genese reflektiert werden müssen. Eine rein produktorientierte Perspektive betrachtet allein fertige Software-Artefakte, die als statische Strukturen begriffen werden und sich nicht aus dem Entstehungsprozess ergeben.[13] Die prozessorientierte Sicht hingegen lenkt den Fokus

[11] An dieser Stelle ist die Frage nach der Zuverlässigkeit (*trust*) solcher Informationen bzw. den Möglichkeiten der Manipulation zu stellen. Diesen durchaus berechtigten Einwand möchte ich jedoch vorläufig unbeachtet lassen, da der Schwerpunkt der Untersuchung auf dem Entstehen von Struktur liegt.
[12] Das Konzept der partizipativen Software-Entwicklung ist als Gegenentwurf zu dem auf die Fertigstellung fehlerfreier Endprodukte ausgerichteten „Wasserfall"-Modell zu verstehen.
[13] "The **product-oriented** perspective regards software as a product standing on its own, consisting of a set of programs and related defining texts. In doing so, the product-oriented perspective abstracts from the characteristics of the given base machine and considers the usage context of the prodcut to be fixed and well understood, thus allowing software requirements to be determined in advance." (Floyd 1987:194)

verstärkt auf Entwicklungsprozesse im Einsatzumfeld solcher Artefakte. Software wird nicht mehr nur als ein fertiges Endprodukt beschrieben, sondern in einem dynamischen Begleitprozess ständig kontrolliert, revidiert, verbessert und ausgebaut. Doch auch solche Prozesse allein sind nur von flüchtiger Erscheinung, wenn sie sich nicht zu Artefakten verhärten.[14] Das eigentliche Potenzial der partizipativen Softwareentwicklung besteht nun in der Verknüpfung der Produkt- mit der Prozessebene. Statt fertiger Programme gibt es Versionen, die in wiederkehrenden, aufeinander aufbauenden Designzyklen zu einem Produkt reifen und wiederum den folgenden Revisionsprozess beeinflussen. Produkt und Prozess sind nach dieser Sicht systematisch aufeinander bezogen; Struktur ergibt sich aus der Gesamtheit aller beteiligten Praxen.[15]

4.3 Suchmaschinen

Auch seitens der Medientheorie wird die Frage thematisiert, wie Technologien auf gesellschaftliche Bedürfnisse reagieren können bzw. wie umgekehrt Veränderungen im Sozialen die Entwicklung von Technik beeinflussen. Ausgehend von dieser Fragestellung entwickelte Hartmut Winkler ein Untersuchungsmodell[16], das es ermöglicht, solche wechselseitigen Beziehungen von Praxen und Struktur zu beschreiben. Hierzu überträgt er Erkenntnisse aus Sprachwissenschaft und kognitiver Psychologie auf Adaptionsprozesse im Bereich der Medientechnologien.[17] Auf Basis der zwei „Seinsweisen" von Sprache, dem *Sprechen* einerseits sowie dem *System* andererseits, äußert sich die Komplementarität von „Resultat" und „Nutzen" nun darin, dass sich bestimmte Strukturen im System durch häufigen Gebrauch verstärken, andere hingegen abgeschwächt werden. Sprechakte („Praxen") tragen zum Aufbau der Gesamtstruktur bei, umgekehrt schöpft der einzelne Sprecher zur Gestaltung der Rede immer aus dem Gesamtsystem. Entscheidend ist hier, dass sich das semantische System der Sprache allein aus den Sprechakten ergibt; es ist keine andere Möglichkeit der Verdichtung gegeben:

> Ob ich eine vorfindliche Technik autonom setze und deren Wirkungen auf den sozialen Prozess untersuche oder ob ich darauf beharre, daß die Technik selbst ihre Wurzeln in Praxen, im Sozialen oder in kommunikativen Akten hat, - es zeigt allen an, welche Phase des Zyklus ich in den Mittelpunkt meines Interesses stelle: es handelt sich um die jeweils vereinseitigte Behandlung eines Gesamtprozesses, der grundsätzlich Einschreibung und Zurückschreiben, den Übergang von Praxen in Niederlegung und den zweiten Übergang von Niederlegung in Praxen umfasst (Winkler 2002:307).

[14] "The **process-oriented perspective** [...] views software in connection with human learning, work and communication, taking place in an evolving world with changing needs. [...] From the process-oriented perspective, the actual product is perceived as emerging from the totality of interleaved processes of analysis, design, implementation, evaluation and feedback, carried out by different groups of people involved in system development in various roles." (Floyd 1987:194)

[15] "In keeping with this idea, I had originally used the notions of "established" vs. "revised" point of view for naming these perspectives. [...] The two perspectives actually coexist in time; they are to some extent complementary, and no individual known to me argues from one of these perspectives only. The change would not imply giving up one of these perspectives in favour of the other but rather improving our understanding of how we should allow them to interact." (Floyd 1987:193)

[16] Einen ersten Ausblick auf die Frage, inwieweit sich „Akte des Gebrauchs" in der Technik niederschlagen, gibt Winkler in *Docuverse – Zur Medientheorie der Computer* (1997a:338); aufgegriffen und weitergeführt wurde die Fragestellung in verschiedenen Aufsätzen in den Folgejahren, insbes. in Winkler (1997b). Explizit formuliert wurde das Modell schließlich in Winkler (2002).

[17] Es ist möglich, so Winkler, von einem anthropologisch orientierten Standpunkt aus, die Menschen und ihre Aktionen, Handlungen in den Mittelpunkt zu stellen und so die Entwicklungsgeschichte der Medien als einen „fluiden Diskurs" zu begreifen. Oder aber man betrachtet von einem technikzentrierten Standpunkt aus die Schrift, Technik, die Artefakte selbst als „materielle Niederlegungen", geronnene Strukturen, inmitten eines gesellschaftlich-technischen Environments. Das Modell geht der Frage nach, „auf welche Weise Diskurse ihre Kontinuität organisieren." (Winkler 2002:298f.)

Dieses sehr allgemeine Modell ist durch die zyklische Verbundenheit von Praxen und Struktur gekennzeichnet und ist, so Winkler selbst, auf viele Anwendungskontexte übertragbar.[18] Semantik wird allein aus Praxen gewonnen, sozusagen als „Kumulat" geronnener Niederlegungen, die in Struktur umgeschlagen sind (Winkler 2002:308).

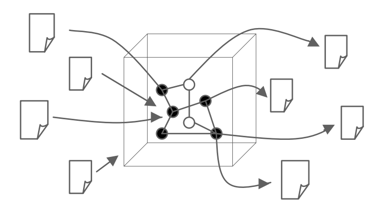


Abb.: Anpassung durch Strukturveränderung
[in Anlehnung an Winkler (2002)]

Geht es nun darum, Strukturannahmen bezüglich des Internet zu treffen, ist dasselbe Modell zugrunde zu legen.[19] Hartmut Winkler beschäftigt sich hier mit den Suchmaschinen, die bei ihrer Entstehung in den Jahren 1994/95 ebenso eine Reaktion auf ein Wissensproblem darstellen. Dieses führt der Autor darauf zurück, dass die Menge der im Netz verfügbaren Texte innerhalb kürzester Zeit exponentiell gewachsen und der Zugriff auf einzelne Dokumente nicht mehr ohne weiteres möglich war (Winkler 1997b:185). Von Interesse sind hier vor allem die im Alltagsgebrauch üblichen Suchmaschinen vom Typ *Google* oder *Altavista*[20], bei denen jeder einzelne Begriff eines durchsuchten Textes in einen Index aufgenommen wird und dann als Suchbegriff zur Verfügung steht.

Die Suchmaschinen werden nun von Winkler als eine „Vertretung der Sprache im Netz" (Winkler 1997b:192) bezeichnet. Abgesehen von einer gewissen „Blindheit im Umgang" mit ihnen und der Tatsache, dass sie eine zentrale Stellung im Netz okkupieren, stellen sie, so Winkler, eine dem System der Sprache (der *langue*) vergleichbare Instanz dar. Ihre besondere Qualität liegt darin begründet, dass sie eine Struktur bilden, von der die Gesamtheit der im Netz abgelegten Texte zunächst unberührt bleibt. Suchmaschinen ermöglichen die nutzerfreie Beobachtung der Aktivitäten und repräsentieren die Gesamtheit der Texte in strukturierter Form; das Resultat eines Suchergebnisses bzw. das Ranking der Treffer beeinflusst zudem den weiteren Verlauf der Navigation.

Unabhängig vom genutzten Typus wirken sich häufige Zugriffe auf Suchmaschinen implizit auf die Struktur des gesamten Netzes aus; sie führen ebenso zum Umbau, wie einzelne Sprechakte zum Umbau des Systems „Sprache" führen. Die einzelne Aktion eines Nutzers im Netz verhalte sich zur Suchmaschine so wie in der Sprache eine einzelne Äußerung zum Gesamtsystem: Die Suchmaschinen geben somit die eigentliche Struktur der Texte vor, die durch die Texte selber entsteht, gleichermaßen als Ergebnis eines Abstraktionsprozesses.

[18] „Dieses schlichte Modell scheint mir ausgesprochen weitreichend zu sein. Und dies ist der Grund, warum ich einige Kraft darauf verwende, ihm innerhalb der Medienwissenschaft Geltung zu verschaffen. Es ist in der Lage, Fragestellungen der Medienwissenschaft, der Kulturtheorie, der Semiotik, der Techniktheorie, der Psychoanalyse und einiger anderer Subdiskurse in systematischer Weise aufeinander zu beziehen." (Winkler 2002:306)

[19] Der Aufsatz über die Suchmaschinen (Winkler 1997) geht zwar zeitlich dem „Modell" (Winkler 2002) voraus; jedoch liegen hier bereits implizit dieselben, später explizit formulierten, Strukturannahmen zugrunde.

[20] Insgesamt werden drei Arten von Suchmaschinen unterschieden: Neben den alltagsüblichen Suchmaschinen, die die abgelegten Texte analysieren, gibt es die hierarchisch sortierten Stichwortsammlungen in der Art von *Yahoo!*. Als dritte Kategorie können die so genannten „semantischen" Suchmaschinen unterschieden werden, die sich das Metadatenkonzept des *Semantic Web* zunutze machen (vgl. Winkler 1997b:187ff.).

Meine Hypothese ist nun, dass es anhand der Suchmaschinen ebenso möglich sein müsste, Semantik als eine dynamische Struktur zu beschreiben, von der die expliziten Auszeichnungsverfahren unberührt bleiben. Parallel zum *Semantic Web* könnte somit noch eine weitere Struktur im Internet gedeihen, die aus den vielfältigen Navigationsprozessen oder Suchaktivitäten resultiert. Zu vermuten ist auch, dass die Nutzerbewegungen zur Architektur des Gesamtnetzes beitragen und einen signifikanten Umbau bewirken könnten. Wozu bedarf es somit noch einer expliziten Form der semantischen Auszeichnung, wenn Semantik schon in der Struktur selber liegt?

5. Was zu tun bleibt...

Die vorgestellten Konzepte zur Erschließung von Bedeutung mittels Strukturbeobachtungen erfordern noch eine weitergehende Betrachtung und Verfeinerung. Es mag vielleicht ein wenig gewagt erscheinen, die weitgehend autonomen und in unterschiedlichen Kontexten geführten Diskurse aufeinander zu beziehen; dennoch ist sowohl dem Ansatz aus der Softwaretechnik als auch dem Modell aus der Medientheorie eines gemeinsam: das Moment der Abstraktion. Produkte setzen den dynamischen Entstehungsprozess und diverse Einbettungszusammenhänge zwangsläufig voraus, ebenso wie Strukturen notwendigerweise auf Praxen („Akte der Einschreibung“) zurückgehen müssen. Neu an diesem Hybridmodell ist, dass die Aktivitäten des Benutzers, in Algorithmen implementiert, auf Technik einen gewissen Anpassungsdruck ausüben und zu Veränderungen führen können. Nicht mehr die anthropomorphisierende, dualistische Sicht auf Mensch und Maschine steht im Mittelpunkt, sondern Gebrauch und Technik sind zyklisch miteinander verwoben.

Diskutierbar bleibt jedoch, um welche Art von Semantik es sich überhaupt handelt, denn: Können Technologien (wie das Internet, Softwareprodukte oder -systeme) überhaupt in denselben Kategorien beschrieben werden wie die natürliche Sprache? Im Gegensatz zu einer natürlichsprachlichen Semantik ist es in einer formalen Semantik nie möglich, dass eine Maschine bzw. ein Automat sich aus sich selbst weiterentwickelt bzw. von selbst Bedeutung schöpft. Aus diesem Grund wird auch das Such- bzw. Navigationsverhalten des Benutzers mit einbezogen, das die Dynamik geistiger, schöpferischer Prozesse reflektiert.

Grundsätzlich ist zu klären, wo die Grenzen eines solchen beobachtenden Ansatzes liegen. Ist es also möglich, Eingeständnisse an eine autonome eigendynamische Entwicklung der Medien zu machen, die sich dem unmittelbaren Bewusstsein entzieht und einem gewissen Anpassungsdruck von außen ausgesetzt ist? Wie komplex sind die entstehenden Strukturen? Wie laufen überhaupt Konventionalisierungsprozesse im Medium „Internet“ ab, bei denen z. T. menschliche, z. T. maschinelle Akteure fungieren?

Der vorgestellte Ansatz ist in der Medienwissenschaft positioniert und untersucht das Problem des Wissensmanagement aus kulturwissenschaftlicher Perspektive. Jedoch ist ein hohes Maß an interdisziplinärer Betrachtung erforderlich, da durch den zunehmenden Einsatz interaktiver Medien gegenwärtig weitaus mehr Möglichkeiten zur Verwaltung von Wissen und damit auch zur Bedeutungserschließung gegeben sind als es traditionell möglich war.

6. Literaturverzeichnis

- Andersen, Peter Bøgh: A Theory of Computer Semiotics. Semiotic approaches to construction and assessment of computer systems. Cambridge: Cambridge University Press, 1990.
- Berners-Lee, Tim: Weaving the web. The Past, Present and Future of the World Wide Web by ist Inventor. London: Texere, 2000.
- Berners-Lee, Tim; Hendler, Jim; Lassila, Ora: The Semantic Web. In: *Scientific American* vom 17. Mai 2001. (http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21)
- Dourish, Paul; Chalmers, Matthew: Running Out of Space: Models of Information Navigation. In: *Proceedings of HCI '94*. Glasgow, 1994.
- Dreyfus, Hubert L.: Die Grenzen künstlicher Intelligenz. Was Computer nicht können. Königstein: Athäneum, 1984. (OA., am.: 1972)
- Floyd, Christiane: Outline of a paradigm change in software-engineering. In: Bjerkness, G.; Ehn, P.; Kyng, M. (ed.): *Computers and Democracy. A Scandinavian Challenge*. Aldershot et al.: Avebury, 1987, 193-210.
- Geroimenko, Vladimir: The XML Revolution and the Semantic Web. In: Geroimenko, Vladimir; Chen, Chaomei (eds.): *Visualizing the Semantic Web. XML-based Internet and Information Visualization*. London: Springer, 2003, 3-14.
- Haugeland, John: Künstliche Intelligenz - Programmierte Vernunft? Hamburg: McGraw-Hill, 1987. (OA., am.: 1985)
- Kamphusmann, Thomas: Zeichen in Wissensmanagmentsytemen. In: K. Bauknecht et al. (Hrsg.): *Informatik 2001 – Tagungsband der GI/OCG-Jahrestagung*, Bd. II, Österreichische Computergesellschaft 2001, 743-747.
- Keil-Slawik, Reinhard: Zwischen Vision und Alltagspraxis: Anmerkungen zur Konstruktion und Nutzung typographischer Maschinen. In: G.G. Voß, W. Holly und K. Boehnke (Hrsg.): *Neue Medien im Alltag: Begriffsbestimmung eines interdisziplinären Forschungsfeldes*. Opladen: Leske & Budrich, 199-220.
- Minsky, Marvin: Mentopolis. Stuttgart: Klett-Cotta, 2. Aufl. 1994. (OA., am.: 1985)
- Nake, Frieder: Das algorithmische Zeichen. In: K. Bauknecht et al. (Hrsg.): *Informatik 2001 – Tagungsband der GI/OCG-Jahrestagung*, Bd. II, Österreichische Computergesellschaft 2001, 736-747.
- Ortner, Erich: Missverständnisse aus der Informatik – erörtert von einem konstruktivistischen, nicht von einem empiristischen Standpunkt. In: K. Bauknecht et al. (Hrsg.): *Informatik 2001 – Tagungsband der GI/OCG-Jahrestagung*, Bd. II, Österreichische Computergesellschaft 2001, 755-762.
- Peirce, Charles S.: Phänomen und Logik der Zeichen. Frankfurt/Main: Suhrkamp, 1983.
- de Saussure, Ferdinand: Cours de linguistique générale. Publié par Charles Bailly et Albert Séchehaye. Paris: Éditions Payot & Rivages, 1967. (OA.: 1916)

- Schefe, Peter: Künstliche Intelligenz – Überblick und Grundlagen. Grundlegende Konzepte und Methoden zur Realisierung von Systemen der künstlichen Intelligenz. Mannheim: Bibliographisches Institut, 1986.
- Schefe, Peter: Die Rolle der Ontologie in der Softwaretechnik am Beispiel der Visualisierung. In: K. Bauknecht et al. (Hrsg.): *Informatik 2001 – Tagungsband der GI/OCG-Jahrestagung*, Bd. II, Österreichische Computergesellschaft 2001, 788-793.
- Winkler, Hartmut: Docuverse. Zur Medientheorie der Computer. Regensburg: Boer, 1997.
- Winkler, Hartmut: Suchmaschinen. Metamedien im Internet? In: Becker, Barbara; Paetau, Michael (Hrsg.): *Virtualisierung des Sozialen. Die Informationsgesellschaft zwischen Fragmentierung und Globalisierung*. Frankfurt/Main: Campus-Verlag, 1997, 185-202.
- Winkler, Hartmut: Das Modell. Diskurse, Aufschreibesysteme, Technik, Monumente – Entwurf für eine Theorie kultureller Kontinuierung. In: Pompe, Hedwig; Scholz, Leander (Hrsg.): *Archivprozesse. Die Kommunikation der Aufbewahrung*. Köln: Dumont, 2002, 297-315.

Ontologie-Konstruktion
am Beispiel von XML TopicMaps

Dr. Johannes Busse, www.jbusse.de
Erziehungswissenschaftliches Seminar, Universität Heidelberg
aktuelle Textfassung:http://www.jbusse.de/archiv/OntologieMarburg2003.html

Zusammenfassung

Der Beitrag diskutiert eine der leitenden Fragestellungen des Symposiums: *Ist [...] die Objektorientierung ein unbestrittenes Paradigma - spiegelt sie die "natürliche" Weltsicht [...] wider?* Folgende Thesen dienen dem Einspruch:

These 1: Die Informatik kennt viele formale Notationen. Diese ergänzen sich eher fruchtbar als dass sie sich wechselseitig ausschließen. OO ist daher kein Paradigma im starken (z.B. Kuhn'schen) Sinn.

These 2: Modellierung spiegelt nicht eine spezifische Sicht auf "die" Welt wider, sondern erzeugt sie erst. Dies wird an einem konkreten Anwendungsfall auf Basis der Notation XML Topic Maps (XTM) plausibel gemacht und gilt auch für OO.

These 3: Es ist eher die formale Notation des Modellierungsansatzes als die dahinter liegende "große" Idee, die eine Ontologie des Anwendungsbereichs erzeugt.

ad 1: Zur "Paradigmen"-These der Veranstalter

In der Ausschreibung zu dem Symposium stellen die Veranstalter folgende -- prima facie verführerische! -- Frage:

- "Ist [...] die Objektorientierung ein unbestrittenes Paradigma?"

Unbestritten ist die Objektorientierung (OO) eine in der Informatik derzeit äußerst bedeutsame Modellierungstechnik. Sie stellt für uns Informatiker weitreichende Schemata nicht nur für das *engineering* von Software, sondern auch für das Denken und Erkennen -- wahr-nehmen im Wortsinn -- bereit. In diesem weiten Sinne ist OO ein Paradigma (nicht unbedingt nur, aber auch) der Informatik.

Sucht man allerdings den mit schärferen begrifflichen Werkzeugen geführten Diskurs, so müsste der Begriff des Paradigmas um einiges präziser gefasst werden. Hierfür bietet sich etwa das in der Epistemologie und Wissenschaftsforschung bedeutsame Begriffsverständnis von Thomas Kuhn (1962, 1976) an, demgemäß Paradigmen durch blinde Flecke, unhinterfragte Glaubensannahmen und selbsterfüllende Prophezeiungen der Wissenschaft gekennzeichnet sind, die zu letztlich inkommensurablen Theoriebildungen führen.

In der Informatik breit konsensfähig dürfte die Beobachtung sein, dass die Informatik über viele verschiedene, ihren Gegenständen jeweils unterschiedlich angemessene Modellierungstechniken verfügt: Jede Programmiersprache, jeder Modellierungsansatz birgt -- kognitiv, für uns Menschen, nicht für die Maschine! -- seine eigene Denkweise in sich.

Meiner subjektiven Einschätzung gemäß ist dies gut so, weil erst eine solche Perspektivenvielfalt Erkenntniswege öffnet und auch eine dem Einzelfall angemessene Modellierung ermöglicht: Denn wer lediglich eine Kultur kennt, kennt auch diese nicht. Ein guter Informatiker ist sein -- oder ihr -- eigener Souverän über eine Vielfalt von Denkweisen. Informatik zu studieren heißt deshalb, sich immer wieder neue Denkweisen zu erschließen. Jeder, der zum ersten mal mit einem deklarativen oder einem funktionalen Denkmodell an ein Problem herangegangen ist konnte die Erfahrung machen, welche "Knoten in Hirn" dazu zunächst einmal zu lösen waren.

Wo aber eine solche Vielfalt in Ausbildung und Praxis tatsächlich kultiviert wird, hält OO keinem starken Paradigmenbegriff im Sinne Kuhns stand. Auch wenn eine OO-Modellierung tatsächlich gewisse Theorie-"imperialistische" Eigenschaften aufweisen kann muss sie sich insgesamt doch nicht so exklusiv geben, dass sie andere Modellierungsansätze ausschließt oder gar un-denkbar macht. Im Gegenteil sind durch Prinzipien wie Abstraktion und Kapselung Strukturen geschaffen, die zumindest "innerhalb" (sic!: eine typische Ontologie erzeugende Metapher!) einzelner Objekte auf Methoden-Ebene nahezu beliebige Programmiersprachen eingesetzt werden können.

Nur wenn Informatik zu betreiben immer auch OO heißen würde, könnte OO im starken Sinn als "der" paradigmatische Kern der Wissenschaft Informatik bezeichnet werden.

ad 2: Die "Spiegelungs"-These der Veranstalter

In enger Nähe zur Paradigmenthese geben die Veranstalter weiterhin zu bedenken:

- "Spiegelt sie [die OO] die 'natürliche' Weltsicht [...] wider?"

Offensichtlich entscheidet sich die Zustimmung oder der Widerspruch zur These auch hier an Begriffen, in diesem Falle dem Naturbegriff. Gemäß dem eingeführten, unkritischen Sprachgebrauch könnte man "natürlich" übersetzten als etwas, was sich "so anbietet wie es ist", also "wirklich" ohne Maske, "unverborgen". Das Unverborgene aber ist auf griechisch die Aletheia, die heute gemeinhin mit "Wahrheit" übersetzt wird. Es ist nach nicht weniger als der Wahrheit von OO gefragt.

Es ist tatsächlich richtig und wichtig, die Wahrheitsfrage auch außerhalb der akademisch betriebenen Philosophie immer wieder neu vor der Folie des eigenen wissenschaftlichen Tuns zu thematisieren. Denn nur der Wissenschaftler, der sich gegenüber der schieren Größe und Unbewältigbarkeit des Faches Wahrheitstheorien (Einführung siehe z.B. Puntel 1993, Skirbekk 1989) nicht all zu befangen zeigt, vermag die für jede Wissenschaft unabdingbaren wissenschaftstheoretischen Bezüge auf sein eigenes Fach zu beziehen.

Wenn diese Thematik hier also in Gestalt der Spiegelungs-These zur Diskussion gestellt wird, ist zunächst vor der naiven Weltsicht zu warnen: dort die Realität? hier wir, die wir sie erkennen? Schon Plato destruiert in seinem Höhlengleichnis diese scheinbar so klare Objekt-Subjekt-Beziehung, deren Verhältnis auch in der nachfolgenden 2400-jährigen Philosophiegeschichte immer wieder neu zu bestimmen versucht wird -- auch in Zusammenhang mit dem Naturbegiff, sich bei genauerem Hinsehen historisch als ein überhöhtes Konstrukt der Romantik des 19. Jahrhunderts in Literatur und Malerei entpuppt: Unberührte Landschaft war damals schlichtweg Ödland, als Natur galt dagegen eine wohl bestellte und hohen Schönheitsidealen verpflichtete Kulturlandschaft. Natur war und ist etwas Gemachtes. Daher die (nicht besonders aufregende) *These*: Wie jeder andere Modellierungsansatz auch spiegelt OO

nicht Welt, sondern bewirtschaftet in ihr vom Menschen gemachte informationstechnische Kulturlandschaften.

An dieser Stelle philosophisch weiter in die Tiefe zu gehen würde einen abstrakten Streit um Begriffe nähren. An einem solchen kann uns in unserem interdisziplinären Symposium aber nicht gelegen sein.

ad 3: Jede einzelne formale Notation erzeugt eine spezifische Ontologie

Ich will im folgenden an einem konkreten Anwendungsbeispiel eine kleine, spezifische Hypothese deutlich machen (ohne sie freilich in einem starken Sinn nachweisen zu können) und auf ihre mögliche Tragweite abklopfen. Kontext ist die offensichtlich konsensfähige Beobachtung, dass jedes Programmierparadigma eng mit einem eigenen Denkmodell zusammenhängt. Meine spezifischere Hypothese lautet: *Jeder einzelne konkrete Formalismus (Programmiersprache, formale Notation, Modellierungsstandard) erzeugt seine eigene spezifische Ontologie*.

Ein Modellierungsbeispiel

Am folgenden Anwendungsfall soll verdeutlicht werden, was mit der Hypothese gemeint ist:

In einem erziehungswissenschaftlichen Projekt zum Einsatz Neuer Medien in der Lehre (http://www.studbene.de/) lernen Studierende der Sozial- und Verhaltenswissenschaften, ihre eigenen Texte (Hausarbeiten, Lehrmaterialien, Arbeitsblätter) direkt in xhtml zu codieren. Innerhalb der xhtml-Dateien sollen dabei alle Textabschnitte, die als in sich geschlossene semantische Einheit interpretierbar sind, eindeutig adressierbar gemacht werden. Technisch wird dies durch den xhtml-Tag `div` gelöst:

```
<div id="eindeutigeID">
   ... hier beliebiger in xhtml codierter Inhalt ...

   <div id="ZeiteEindeutigeID">
       ... der auch verschachtelt sein darf
   </div>

</div>
```

Die spannende Herausforderung besteht darin, das semantische Netz der "Querbezüge" zwischen solchen Textscheiben zu modellieren.

Eine erste Modellierung dieses semantischen Netzes von Querbezügen orientiert sich an einem klassischen Relationen-Modell aus der relationalen Datenbanktechnik, bei der die semantischen Querbezüge als geordnete 3-Tupel in einer Tabelle etwa wie folgt abgelegt werden:

Tabellenkopf: `(Querbezug, Quelle, Ziel)`

Instanz ("Zeile"): `("Schriften", "JohannesBusse", "3-514-317665-X")`

Aus strategischen Gründen hatte man sich in dem o.a. Anwendungsfall nicht für eine relationale Modellierung, sondern für eine Modellierung auf Basis des Standards *XML Topic Maps* (XTM) entschieden. In diesem Standard werden Querbezüge zwischen "Topics" als "Assoziationen" dargestellt, in denen die beteiligten Topics in "Rollen" schlüpfen. In der für XML typischen (bisweilen etwas enervierend expliziten) Ausdrucksweise wird die oben genannte Instanz eines Querbezugs wie folgt notiert:

```
<association>

   <instanceOf>
      <topicRef xlink:href="#Schriften"/>
   </instanceOf>

   <member>
      <roleSpec>
         <topicRef xlink:href="#Quelle"/>
      </roleSpec>
      <topicRef xlink:href="#JohannesBusse"/>
   </member>

   <member>
      <roleSpec>
         <topicRef xlink:href="#Ziel"/>
      </roleSpec>
      <topicRef xlink:href="#3-515-317665-X"/>
   </member>

</association>
```

Was lässt dieses Beispiel vermuten?

In einigen Einführungen in die XTM-Philosophie wird der oben exemplarisch aufgeführte Datensatz als XML-Darstellung der o.A. Zeile erläutert (z.B. Obrst 2003). Weil im Gegensatz zur relationalen Tupel-Notation in der XML-Notation keine eindeutige Reihenfolge der `member`-Elemente mehr definiert ist, müssten, so die Erläuterung, die Zuordnungen von Tabellenkopf und Zeileninhalt eben explizit notiert werden.

Diese Erläuterung ist nicht falsch und muss aus didaktischer Sicht als sehr gelungen bezeichnet werden. Und gerade deshalb wird dabei leicht übersehen, dass die XML-Notation etwas bietet, das in der relationalen Denkweise so noch nicht enthalten ist. Syntaktisch ist aus dem 3-Tupel ein 5-Tupel geworden mit der (beabsichtigten? zufälligen?) Nebenfolge, dass an der Stelle der `roleSpec` nun *beliebige* Topics eingesetzt werden können. In der Denkweise der relationalen Datenbanktechnik würde dies einem System entsprechen, in dem jede Definition einer Relation (anschaulich: jeder Tabellenkopf) jederzeit zur Laufzeit um beliebige Werte ergänzt werden könnte (wobei innerhalb der Tabellenzeilen die undefinierten Einträge großzügig etwa mit `undef` aufgefüllt werden könnten). Eine solche "Definitionsfreiheit" sprengt offensichtlich nicht nur die Technik, sondern auch die Philosophie (und die Ontologie?) der relationalen Datenbanktechnik.

Im obigen Beispiel könnte also statt "Quelle" auch eine der Rollen "Autor", "Ko-Autor", "Herausgeber", "Projektleiter" etc. stehen; statt "Ziel" würde man "Aufsatz", "Diskussionsbeitrag", "Abbildung" etc. verwenden. Die Bedeutung des Assoziations-

namens "Schriften" entwickelt sich hin zu einem generalisierten Verständnis. Eine gute XTM-Modellierung degradiert die Rollenbezeichner `roleSpec` also nicht (so wie hier im Beispiel mit "Quelle" und "Ziel" geschehen) lediglich zu einer formalen, d.h. der Notation geschuldeten Information, sondern gibt ihnen materiales, d.h. im modellierten Gegenstand verankertes semantisches Gewicht.

Diese Reichhaltigkeit der XML-Syntax kann im XTM-Standard nun durch folgende Norm zur Tugend gemacht werden: *Objekte stehen nicht mehr lediglich miteinander in Beziehung, sondern sie füllen dabei immer auch eine bestimmte Rolle aus.* Diese Norm kann nicht nur als Aussage über Syntax, sondern auch als eine ontologische Aussage verstanden werden. Als Rolle aufgefasst werden zu können ist damit eine wesentliche Eigenschaft von bestimmten Dingen der Welt.

Dieses Rollenkonzept ist m.E. ein Ort, an dem sich die spezifische Ontologie einer XTM-Modellierung entfaltet und sich von "reinen" OO- (resp. UML-) Modellierungen unterscheidet. Diese Reichhaltigkeit der in XTM darstellbaren Semantik der Kanten-Typen macht eine spezifische Ontologie möglich, und stärker: *erzeugt* sie auch. Was kann damit gemeint sein?

Wie wird eine Weltsicht erzeugt?

Gemäß der oben vorgebrachten kleinen Hypothese soll nun (ebenfalls exemplarisch) plausibel gemacht werden, dass und wie diese spezifische Reichhaltigkeit der Beschreibung zu einer veränderten Wahrnehmung (oder gar Konstruktion?) von "Welt" führen kann. Dies ist ein psychologische Annahme, die auf Basis eines Einzelfalls anhand eines (hypothetischen) Experteninterviews ausgeführt werden soll:

[...] Ja, mit XTM setze ich mich intensiver seit einem halben Jahr auseinander. Ich modelliere experimentell verschiedene Anwendungsbereiche auf Basis des XTM-Standards. [...] Vorwissen bringe ich aus meiner vor mehreren Jahren gehaltenen Vorlesung "relationale Datenmodellierung" mit. UML verwende ich als grafische Notation, jedoch ohne tägliche *hands on*-Erfahrung mit Produktivitätstools. [...]

XTM Assoziationen sind für mich spannend, weil ich hier vor relativ kurzer Zeit wieder einmal das bekannte Gefühl hatte, kognitive Knoten lösen zu müssen. Weil das heißt für mich dann immer: ich lerne etwas dazu.

- Früher, in meiner "Prä-XTM-Phase, gab es -- etwas verkürzt dargestellt -- Objekte (mit Attributen und Methoden), die mittels Relationen miteinander in Bezug gesetzt werden konnten.
- Heute gibt es immer noch Objekte (mit Attributen und Methoden). Die können jedoch dann und nur dann miteinander in Beziehung gesetzt werden, wenn sie bestimmte Rollen einzunehmen bereit sind.

Früher hatte ich viele Dinge in der Welt als Attribute von Objekten "erkannt", eher unabhängig von ihrem Kontext. Heute interpretiere ich diese als Rollen. Wie bei einem Theaterstück! Immer bin ich dann auch sogleich dabei, die anderen beteiligten Charaktere und ihre Rollen zu suchen. [Pause] Denn solche Rollen sind ja immer von ihrem Kontext abhängig. Viele Entitäten, die ich früher noch als isolierte Eigenschaften betrachtet habe, muss ich jetzt als Teil eines komplizierten Geflechts darstellen. Das ist für mich das Interessanteste an XML Topic Maps [...]

[Anm. d. Verf.: Dieser Text wurde von J.Busse als Texttyp "Interview" auf Basis seines internen Lern-Journals konstruiert, um auf diese Weise die hermeneutische Distanz zu seiner eigenen Individualempirie herzustellen. Nicht das Interview selbst, wohl aber sein Inhalt sind hoch authentisch.]

Abstrahierende Interpretation: Der Interviewpartner beschreibt hier einen qualitativen Sprung. Zu seinem Repertoire der formal-strukturellen Weltbeschreibung tritt die ontologische Kategorie der "Rolle" hinzu. Er geht auf den metaphorischen Gehalt des Rollenbegriffs ein, der offensichtlich stets auch die Suche nach geeigneten Gegenrollen einfordert. Auch der Modellierungstyp ändert sich dadurch: eine eher an einzelnen, isolierten, autonomen Objekten orientierte Modellierung wird ergänzt durch eine Modellierung, die auch den Kontext, die "Zwischenräume" zwischen Objekten durch das Rollenkonzept vergleichweise hoch strukturiert. Diese Sichtweise wird vom Interviewpartner selbst der Auseinandersetzung mit dem XTM-Standard zugeschrieben.

Die "Notations"-These als Gegenentwurf

Das Interview und seine Interpretation enthält bereits die vergleichsweise weitgehende These: Nicht die abstrakte Idee wie z.B. "Topic Map", sondern erst die aktive Auseinandersetzung mit dem zugehörigen konkreten W3C-Standard XTM erzeugte hier auf kognitiver Ebene das beschriebene Rollenkonzept. Verallgemeinert: nicht die diffusen "Philosophien" wie "OO", "semantisches Netz" etc., sondern die zugehörigen Formalismen (formalen Sprachen, semi-formalen grafische Notationssysteme) sind es, die in ihrer *Anwendung* (d.h. der Modellierung in der Software-Praxis) eine spezifische Weltsicht erzeugen. Oder knapp: Große Ideen alleine sind "weich", erst der "harte" Formalismus erzeugt die Ontologie.

Diese These wird gestützt durch die Voraussetzungen und Thesen der vorangehenden Abschnitte. Sie sollen zu einem besseren Verständnis zusammenfassend explizit benannt werden:

These 1:

- Die Informatik stellt eine Vielzahl von Modellierungsansätzen bereit; OO ist davon zwar ein wichtiger, jedoch weder der einzige noch unbedingt der zentrale Ansatz. Insbesondere ist OO kein Paradigma im Sinne Kuhns.

These 2:

- Die verschiedenen Modellierungsansätze der Informatik repräsentieren verschiedene kognitive Herangehensweisen, die nicht beliebig austauschbar sind. Unsere Ontologie der "Wirklichkeit" ergibt sich aus der Ontologie des von uns jeweils auf einen Weltausschnitt angewandten Modellierungsansatzes.

In der Argumentation sind weiterhin folgende Hintergrundannahmen enthalten:

- "Welt" wird durch die Sprache(n), die wir sprechen teilweise rekonstruiert, teilweise konstruiert. Dabei gilt Wittgensteins Randbedingung: "Die Grenzen meiner Sprache sind die Grenzen meiner Welt".
- (Semi-)formale Notationssysteme und Programmier-"Sprachen" sind (schriftliche) Sprachen von Menschen für Menschen im strengen Sinn.

Damit wird insgesamt gestützt die These 3:

- Damit die Idee OO wirksam werden kann, muss sie sich in einer Sprache oder Notation instantiieren. UML ist eine, jedoch lange nicht die einzige Möglichkeit dazu.

An diesem Punkt angelangt stellt sich ergänzend die Frage, wie sich eine abstrakte Idee und der *Entwurf* zugehöriger Notationssysteme in der wissenschaftlichen informatischen Forschung gegenseitig beeinflussen. Wir tauchen hier tief in das Zentrum erkenntnistheoretischer Grundlagenfragen ein, bis hin zum bekannten Problem des Zusammenhangs von Sprache und Denken. Es ist zu hoffen, dass unser interdisziplinäres Symposium diesen Diskurs auch für die Informatik fruchtbar machen kann.

Literatur

Aroyo, L; Dicheva, D. (Ed.) (2002): Workshop on Concepts and Ontologies in Web-based Educational Systems: ICCE 2002, International Conference on Computers in Education, 3-6 December 2002, Auckland, New Zealand. http://wwwis.win.tue.nl/~laroyo/ICCE2002_Workshop/proc-Workshop-ICCE2002.pdf

Engelmann, Lutz (2002): Methodenkompetenz im Informatikunterricht: Zur Förderung der Methodenkompetenz im Informatikunterricht am Beispiel der Objekt-Attribut-Methode. In: LogIn, S. 30-36

Hofstadler, Alfred (2001): Datenbanken und Wittgenstein: ontologische Vergleiche: Diplomarbeit an der Universität Wien. http://sammelpunkt.philo.at:8080/archive/00000020/01/hofstadler_dipl/inhaltsverzeichnis.html

Kuhn, Thomas S. (1976): Die Struktur wissenschaftlicher Revolutionen: (The Structure of Scientific Revolutions, 1962). Frankfurt: Suhrkamp

Obrst, Leo; Liu, Howard (2003): Knowledge Representation, Ontological Engineering, and Topic Maps. In: Park 2003, S. 103-148

Park, Jack; Hunting, Sam (Ed.) (2003): XML Topic Maps: Creating and Using Topic Maps for the Web. Boston: Addison Wesley

Pepper, Steve (2000): The TAO of Topic Maps. http://www.ontopia.net/topicmaps/materials/tao.html

Puntel, Lourencino B. (1993): Wahrheitstheorien in der neueren Philosophie: eine kritisch-systematische Darstellung. Darmstadt: Wiss. Buchges, 3., um einen ausführlichen Nachtr. erw. Aufl.. Aufl.

Skirbekk, Gunnar (Ed.) (1989): Wahrheitstheorien: eine Auswahl aus den Diskussionen über Wahrheit im 20. Jahrhundert (stw 210). Ffm.: Suhrkamp, 5. Aufl.

Was kann die Philosophie der Informatik bieten?

Peter Janich

Institut für Philosophie der Philipps-Universität Marburg

0 Einleitung

Die Fachwissenschaften denken vor, die Philosophie denkt nach. Die Fachwissenschaften gehen in der Entwicklung von Begriffen, Theorien und Methoden voran; die Philosophie stellt dazu nachträglich Reflexionen an. Sie übt eine Kritik im Prinzipiellen – im philosophischen Sinne von „Kritik“ als Unterscheiden, Abwägen, Urteilen, und „prinzipiell“ im Sinne von Grundlagen oder stillschweigenden Vorannahmen. Das bedeutet aber auch: wo der Informatiker von Software-Entwicklung spricht, redet der Philosoph vom Informatiker, seinen Verfahren, Zielen und Mitteln.

Diese philosophische Reflexion soll hier mit dem Ziel geführt werden, daran zu erinnern, dass wir für unser Reden verantwortlich gemacht werden, und zwar im Alltag, in den Fachwissenschaften (wie der Informatik) und in der Philosophie. Wenn es also im folgenden um einen Beitrag der Philosophie zur Informatik geht, soll auf allen Stufen des Redens das Verhältnis von Wissen und Verantwortung der leitende Gesichtspunkt sein.

In diesem Sinne soll (1) die stillschweigende Hintergrundphilosophie der (meisten) Informatiker wegen ihrer Unverträglichkeit mit unaufgebbaren Alltagsüberzeugungen kritisiert, (2) in ihren Folgen analysiert und (3) mit einem Gegenmittel versehen werden.

1 Die stillschweigende Hintergrundphilosophie der Informatik

Historisch und systematisch verweisen die Quellen der Informatik auf grundsätzliche Entwicklungen, die sich in Mathematik und Physik abgespielt haben. Einerseits der Formalismus in der Tradition D. Hilberts, andererseits der Empirismus in der Tradition physikalischer Geometrieverständnisse (H. v. Helmholtz, A. Einstein) sind die Vorgaben, aus denen sich eine formalistische und kausalistische Grundüberzeugung speist. Bekanntlich wird diese bestimmend für die Philosophie des Logischen Empirismus, deren Rezeption in der semiotischen Theorie von Ch. Morris, der Ausbildung des in der Informatik herrschenden Verständnisses von Zeichen, Theorie und Information, nachweisbar ist.

Hier ist zu berücksichtigen, gegen welche Positionen oder Ansätze das logisch-empiristische bzw. das formalistisch-kausalistische Wissenschaftsverständnis gerichtet war und geblieben ist:

Vor allem war es der Apriorismus Kants als These, dass die Geometrie synthetisch-apriorische Geltung hat; aber auch jede Art von Normativismus (Ethik, normative Wissenschaftstheorie), Instrumentalismus (in der Physik) und Konstruktivismus (in der Mathematik) gehören als Gegenpositionen hierher.

2 Unverträglichkeit mit Alltagsüberzeugungen

Das formalistisch-kausalistische Selbstverständnis ist unverträglich mit einigen Grundüberzeugungen des Alltagslebens. Dazu vier Beispiele:

- „Alibi-Prinzip“: Eine Person kann wohl zu zwei verschiedenen Zeiten am selben Ort, nicht aber an zwei verschiedenen Orten zur selben Zeit sein. Dies ist nicht nur herrschende Überzeugung, sondern wird, etwa bei einem Strafprozess, auch positiv sanktioniert. Niemand könnte seine Verteidigung auf einem Bestreiten des Alibi-Prinzips aufbauen.

  Aber das Alibi-Prinzip gilt weder logisch, noch kann es empirisch widerlegt werden.

- „Subjektivitäts-Prinzip“: Wir kennen im Alltag (wie in den Wissenschaften) eine ungeheure Fülle von Vorgängen, die nur in einer zeitlichen Richtung erlebt werden. Das Weinglas fällt zu Boden und zerbricht, aber es springen keine Glasscherben auf den Tisch und setzen sich zu einem Weinglas zusammen. Von selbst kühlt das Badewasser aus; niemals erwärmt es sich von selbst. Vorgänge dieser Art werden bekanntlich durch den zweiten Hauptsatz der Thermodynamik abgedeckt. Aber diese Reihenfolge sowohl in alltäglichen wie in wissenschaftlichen Aussagen hängen vom Konsens aller Sprecher bezüglich früher und später ab. All diese Beispielsätze könnten nicht semantisch sinnvoll formuliert werden ohne diesen Rückgriff auf die Subjektivität des (in der Kulturgemeinschaft zum Konsens gebrachten) subjektiven Zeit-Erlebens.

- „Normativitäts-Prinzip“: Alle ebenen Körperoberflächen passen (dreh- und verschiebbar) aufeinander. Operativ lässt sich die Ebenheit am Kriterium definieren, dass es zu einer ebenen Oberfläche zwei Paßstücke (etwa Gipsabdrücke) gibt, die auch untereinander (dreh- und verschiebbar) passen.

  Aus dieser Definition mit einem Existenzquantor für die beiden Paßstücke folgt logisch nicht der Allsatz, dass alle Ebenen aufeinander passen. Empirisch geben wir aber den Satz

zur Falsifikation ebenfalls nicht frei, sondern behaupten, im Falle fehlender Passung vermeintlich ebener Oberflächen sei mindestens eine der beiden in Wahrheit nicht eben.

- „Prinzip der methodischen Ordnung“: Bei ungezählten Handlungen des Alltagslebens, des Handwerks, der Technik, der Laborforschung usw. gilt: Der Erfolg (als Erreichen des Zwecks) hängt von der Reihenfolge der Teilschritte ab. Wer über einen Graben springen möchte, wird zuerst anlaufen und dann springen, nicht umgekehrt. Wer Russische Eier zubereiten möchte, wird die Eier zuerst kochen, dann schälen, dann halbieren, dann garnieren. Solche Reihenfolgen im kinetischen, poietischen und praktischen Handeln beruhen weder auf Natur- noch auf Sittengesetzen, sondern sind zweckrational ausgezeichnet.

  Im Alltagsleben weicht de facto niemand von dieser Reihenfolge ab, wenn es um ihre sprachliche Verhandlung geht; sei es im Auffordern (Gebrauchsanweisung, Bauanleitung, Kochrezept), im Behaupten (Erzählung, Zeugenbericht) oder bei anderen Sprechhandlungen wie Fragen, Versprechen, Bekunden usw. Das Prinzip der methodischen Ordnung verbietet explizit diese im Alltagsleben universell anerkannte Praxis: Man weiche im Reden bezüglich der Reihenfolge von Teilschritten nicht von der in der besprochenen Praxis erfolgreichen Reihenfolge ab.

  Der mögliche Einwand gegen solche „Prinzipien“, Alltagsüberzeugungen seien für die Wissenschaften unerheblich, trägt nicht. Wissenschaften sind Hochstilisierungen alltäglicher Praxen, behalten ihren Sitz im Leben, etwa was die Biographie (Sprach- und Handlungsvermögen, Herkunft, Karriere, Ziele) der Wissenschaftler betrifft, und die Lebenswelt ist auch die Bezugsebene für Rechtfertigungsprobleme wissenschaftlicher Forschung. (Hier sei zum ersten Mal erinnert daran, dass wir für unser Reden verantwortlich gemacht werden.)

3 *Kritik des Formalismus*

Im Rahmen der Informatik wird (im wesentlichen zwangsläufig und generell) unterstellt, aus der Syntax ließe sich die Semantik gewinnen. (Für Belege muss auf die Literatur[1] verwiesen werden. Hintergrund ist die behavioristische/kausalistische Umdeutung der Zeichentheorie von Ch. S. Peirce durch Ch. Morris, und die Übernahme durch W. Weaver in der Kommunikations-/Informationstheorie von C. Shannon und W. Weaver (1948/49))

Der hier enthaltene Defekt lässt sich kurz dadurch charakterisieren, dass das klassische Nachrichtenübertragungsmodell (und entsprechend Datenverarbeitungsmodell) einen archimedischen Beobachter außerhalb des Systems benötigt, der entscheidet, ob der Hörer (Nachrichtenziel) den Sprecher (Nachrichtenquelle) „semantisch“ verstanden hat. Im „richtigen Leben“ dagegen, d.h. in tatsächlicher Kommunikation müssen aber die Kommunikationsparteien selbst darüber entscheiden. Das heißt, die Halbierung eines minimalen Kommunikationsschritts auf einen einzigen Durchgang von der Quelle zum Ziel und das unterstellte monologische Sprachverständnis (ohne semantische Kontrolle im Dialog) verhindert die Definierbarkeit von Funktionskriterien für technische Systeme, die nachrichtentechnisch beherrscht werden sollen. Nur im verständigen Dialog, in dem jede der Dialogparteien mindestens einmal den Rollenwechsel vom Hörer zum Sprecher bzw. vom Sprecher zum Hörer absolviert, ergibt sich die Unterscheidung von semantisch relevanten und semantisch irrelevanten technischen Störungen.

Dies gilt bereits für ein einfaches Beispiel wie das handschriftliche Abschreiben eines lateinischen Textes durch einen Mönch, der nur Kopist ist, aber kein Latein versteht: Ob Abschreibfehler den Sinn verändern, kann nur der kundige Lateiner entscheiden.

Für alle von der Informatik behandelten technischen Systeme folgt daraus: Leistungsgleiche technische Substitute für menschliche Sprach- und Kognitionsleistungen haben primär von semantischen Funktionskriterien auszugehen, relativ zu welchen erst im zweiten Schritt die syntaktischen bestimmt werden können. Rein syntaktische Maschinen sind semantisch blind.

4 *Kritik des Kausalismus*

Wie sich der Formalismus als irrtümlicher Versuch charakterisieren lässt, die Semantik aus der Syntax gewinnen zu wollen, so lässt sich der Kausalismus charakterisieren dadurch, dass die Geltung oder Anerkennung aus der Syntax gewonnen werden soll – eben als Kausalwirkung. Im simplen Beispiel: Das Rechenergebnis an einem Taschenrechner soll physikalisch aus dessen Funktion erklärt werden, d.h. kausal impliziert werden.

Nun ist bekanntlich Rechnen nicht das Produzieren irgendwelcher, sondern gültiger Ergebnisse; und diese Leistung soll ein Taschenrechner technisch leistungsgleich substituieren. Würde aus der physikalischen Beschreibung des Rechners definitorisch oder kausal die Wahrheit der Rechenresultate (definiert als Verhältnis von Input- und Outputstellungen des Systems) folgen, müsste umgekehrt bei einem Rechnerdefekt gelten: Falsche Rechenresultate implizieren die Ungültigkeit der Rechnerphysik. (Mein Beispiel: Am Display möge an einer Stelle der Mittelstrich ausgefallen sein, so dass statt einer „8“ eine „0“ erscheint.) Tatsächlich nimmt

niemand an, dadurch sei ein physikalischer Satz über den Rechner widerlegt. Vielmehr wird mit der weiterhin gültigen Rechnerphysik gegebenenfalls der Defekt behoben.

Analog zur Semantik ist damit festgestellt: Die Geltung von Ergebnissen syntaktischer Maschinen ist keine Kausalwirkung ihrer Funktion. Vielmehr ist nach dem Prinzip der methodischen Ordnung erst die Wahrheit von Rechenresultaten (und den Methoden ihres Erreichens) als Zweck (und Explanandum) festzuhalten; dann kann relativ zu diesem Zweck als Mittel (und Explanans) eine Maschine zur leistungsgleichen Substitution des Rechnens erfunden, konstruiert, gebaut und benützt werden. Defekte Maschinen verfehlen lediglich menschliche Zwecke, falsifizieren aber keine so genannten Naturgesetze.

Beide Defekte, der formalistische bezüglich der Semantik und der kausalistische bezüglich der Geltung, lassen sich zurückführen auf die Zeichentheorie von Morris, die einerseits auf einem monologischen Sprachverständnis beruht, statt die unverzichtbare Kontrolle des „Interpretanten“ (als Vermittler von Zeichen und Bezeichnetem) im Dialog zweier Parteien zu berücksichtigen, andererseits auf der behavioristischen Auffassung, dass das (kausale) Reiz-Reaktions-Muster der Interpretant für den tatsächlichen Zeichengebrauch bei Menschen, Tieren und Maschinen ist.

Es ist nicht zu sehen, dass eine Debatte um Software-Entwicklung und die Frage nach ihrer Objekt-, Subjekt- oder Handlungsabhängigkeit derzeit anders geführt wird als unter Rückgriff auf das Zeichenverständnis von Morris (oder von einem auf Morris reduzierten Peirce) durch die (meisten) Informatiker.

5 *Heilmittel*

Ein Ausweg aus den geschilderten Schwierigkeiten der Informatik-Hintergrundphilosophie zeigt sich in der Berücksichtigung der Tatsache, dass wir für unser Reden verantwortlich gemacht werden. Reden ist Handeln, wo Handeln philosophisch verstanden wird als das, was uns von unseren Mitmenschen als Verdienst oder Verschulden zugeschrieben wird.

Es ist der für jedes Mitglied einer Gemeinschaft unverzichtbare Sozialisationsschritt, Handeln in diesem Sinne vom bloßen Verhalten (behavior – nicht zu verwechseln mit conduct, Verhalten im Sinne von Handlungsweisen) zu unterscheiden. Diese Unterscheidung lernen und beherrschen wir, weil kein verantwortlicher Erzieher ein Kind loben oder tadeln würde für Erschrecken, Stolpern, Nießen müssen, für Reflexe und die Prozesse des Stoffwechsels. Anderes wird uns dagegen zugerechnet, bis in die Rolle des Bürgers im demokratischen Rechtsstaat hinein; dort ist es z.B. die juristische Unterscheidung von vorsätzlich, fahrlässig und unschuldig, die auf uns angewandt wird, und die wir im Alltag als absichtlich, versehentlich und unbeteiligt eingeübt haben.

Handlungen unterscheiden sich vom bloßen Verhalten in vielerlei Aspekten: Sie können gelingen und misslingen, Erfolg und Misserfolg haben, d.h. ihren Zweck erreichen oder verfehlen; man kann sinnvoll zu ihnen auffordern, sie in kinetische, poietische und praktische, also Bewegungs-, Herstellungs- und Kommunikationshandlungen einteilen; für die Handlungstheorie muss auf die Literatur verwiesen werden.[2]

Wichtig für die Informatik sind zwei Aspekte: Reden als Kommunizieren mit anderen Menschen erfüllt alle Charakteristika des Handelns. Und Handeln lässt sich, für den Kommunikationsaspekt zentral, nach zwei weiteren Aspekten differenzieren: (1) Handeln kann als Beteiligungshandeln, gemeinschaftliches und individuelles Handeln ausgeführt werden (Beteiligungshandlungen wie „Wettlaufen“ können nur gelingen, wenn sich eine andere Person beteiligt; gemeinschaftliche Handlungen können nur Erfolg haben mit fremder Hilfe: der Zweck, einen schweren Balken zu heben, wird mit einigen Helfern realisiert; und wo weder Gelingen noch Erfolg von anderen Agenten abhängt, handeln wir individuell). Und (2) Handlungen tauchen in der Vollzugs- und in der Beschreibungsperspektive auf, wobei letztere noch einmal die Beschreibungsperspektive eines Teilnehmers oder eines Beobachters sein kann.

Die Kommunikationshandlungen des Redens, in denen es um gegenseitiges Verstehen und Anerkennen geht, sind allesamt Beteiligungs- und Gemeinschaftshandlungen. Wie das (semantische) Verstehen auf Gelingen und Erfolg von sprachlichen Beteiligungshandlungen zurückgespielt werden kann, und wie die Anerkennung (bei Behauptungen die Geltung oder Wahrheit, bei Fragen, Aufforderungen, Bekundungen usw. die entsprechende Anerkennungsform) auf Gelingen und Erfolg von Gemeinschaftshandlungen zurückgespielt werden kann, ist der Literatur zu entnehmen.[3] Hier genügt die Feststellung, dass die Probleme von Semantik und Geltung nach dem Prinzip der methodischen Ordnung pragmatisch mit den Mitteln der Handlungstheorie gelöst sind. Monologisches, formalistisches und kausalistisches Verständnis von Sprache, Kommunikation und Information lassen sich so korrigieren – und dies immer im Blick auf das Ziel, Wissenschaft und Verantwortung zusammenzuhalten und der Einsicht zu genügen, dass wir unser Reden zu verantworten haben. (Hierzu ist keine große Philosophie der Verantwortung erforderlich. Verantworten heißt hier nur in alle Richtungen Antwort geben können, wenn wir nach unseren Handlungen gefragt werden, wiederum in der

gesamten Bandbreite des täglichen Zusammenlebens über Fragen etwa vor einem Gericht bis zur Verteidigung fachwissenschaftlicher und philosophischer Thesen.)

Die distanzierte Beschreibungsperspektive ist dabei durch die Vollzugsperspektive ersetzt, wie wir auch generell und unverbrüchlich zwischen Vollzug und Beschreibung unterscheiden. Oder wer würde nicht den Vollzug eines Mordes von der (teilnehmenden) Beschreibung des Zeugen oder der (beobachtenden) Beschreibung des Gerichtsgutachters unterscheiden? Auch Behaupten, Fragen und Auffordern lassen sich nur als Vollzüge geben. Damit wird sichtbar, dass die oben diskutierten Schwächen der Informatik-Hintergrundphilosophie sich der theoretischen Distanz verdanken, in der das Sprechen als Objekt der Beschreibung, nicht mehr als Vollzug in den Blick genommen ist.

6 *Fazit*

Selbstverständlich kommt auch Informatik als Wissenschaft durch Handeln zustande. Dabei handeln Subjekte, also Personen, die durch ihr Handlungsvermögen im Sinne ihrer Zwecksetzungsautonomie, ihrer Mittelwahlrationalität und ihrer Folgenverantwortlichkeit ausgezeichnet sind. Und selbstverständlich richten sich Handlungen auf Objekte, primär auf andere Personen, sekundär auf Dinge und Geschehnisse, aber auch vielfältig auf anderes wie z.B. wiederum auf Handlungen, wenn nämlich Handlungen geplant oder Handlungen handelnd vorbereitet werden.

Damit erweist sich die Alternative von objekt-, subjekt- und handlungsorientiert für die Entwicklung von Softwaretechnik als wenig überzeugend. Die drei Aspekte sind nicht voneinander zu lösen, sondern nur in methodischer Abhängigkeit voneinander befriedigend zu bestimmen.

Für die Handlungen, die Wissenschaft Informatik zu betreiben, lassen sich die zitierten Probleme vermeiden, wenn immer wieder die Frage nach den Zwecken, den Mitteln, den Folgen und Nebenfolgen gestellt wird. Im Falle etwa von Nachrichtenübertragung, Datenverarbeitung, informatischer Steuerungs- und Regelungstechnik, Prozessierung von logischen oder sprachlichen Problemen usw. ist also nach dem Prinzip der methodischen Ordnung immer zuerst der Zweck explizit festzulegen, um dann die Mittel zu suchen und zu ergreifen. Geht es um kognitive Maschinen im weitesten Sinne, werden deren Funktionskriterien als Kriterien der Störungsfreiheit immer an den menschlichen Leistungen semantischer und gegenseitiger Anerkennungskompetenz gemessen. Oder im Rückblick für erfolgreiche Hard- und Softwaretechnik: Die Bestimmung des Explanandums geht stets der Bestimmung des Explanans methodisch voraus.

Da aber die technische Substitution menschlicher Sprach- und Erkenntnisleistungen nicht formal (definitorisch) oder kausal (explanatorisch) bestimmt ist, sondern wieder durch ein Mittel-Zweck-Verhältnis, hat man zu gewärtigen, dass mancher Zweck durch verschiedene Mittel erreicht werden kann, und manches Mittel für verschiedene Zwecke tauglich ist.

Im Hinblick auf den Grundsatz, dass wir für unser Reden (hier als Informatiker) verantwortlich gemacht werden, eröffnet dieses Wissenschaftsverständnis als zweckrationale Praxis die Gelegenheit, Zwecke zu rechtfertigen, bevor man sie verfolgt.

Literaturhinweise

1 Ch. W. Morris, *Foundations of the Theory of Signs*, Chicago 1938, dt. *Grundlagen der Zeichentheorie*, Frankfurt a.M. 1988.
C. E. Shannon, W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press 1949, dt. *Mathematische Grundlagen der Informationstheorie*, München 1976.
P. Janich, *Information und Sprachphilosophie*, in: J. Mittelstraß (Hrsg.), Die Zukunft des Wissens, Berlin 2000, S. 78-91.
P. Janich, *Die Naturalisierung der Information*, Sitzungsberichte der Wissenschaftlichen Gesellschaft der Johann Wolfgang Goethe-Universität Frankfurt a.M., Bd. XXXVII, 2 (1999), 36 S.

2 P. Janich, *Logisch-pragmatische Propädeutik. Ein Grundkurs im philosophischen Reflektieren*, Weilerswist 2001.

3 P. Janich, *Kommunikation und Kooperation. Zum methodischen Umgang mit Kulturleistungen*, in: P. Janich, Kultur und Methode, München 2003 (im Druck).
P. Janich, *Grundbegriffe und Aufgaben einer methodischen Medienphilosophie der Kommunikation*, in: L. Nagel, M. Sandbothe (Hrsg.), Systematische Medienphilosophie, Berlin 2003 (im Druck).

Philosophische Kritik des essentialistischen Objekt- und Objektivitätsbegriffes

Morteza Ghasempour (Uni. Köln)

Einen bildungstheoretisch motivierten Gedanken von Rousseau, der besagt, ein Kind, das bloß seine Eltern kennt, kenne auch die nicht recht, auf den Wissenschaftsbereich Chemie übertragend, schreibt einmal Lichtenberg: „Wer nichts als Chemie versteht, versteht auch diese nicht recht.“[1] Mehr denn je ist die hier geforderte Bemühung um Interdisziplinarität aufgrund der beträchtlichen Vervielfältigung spezieller forscherischer Betätigungsfelder und zwecks der Verhinderung der daraus unvermeidlich entstehenden Disziplinsolipsismen notwendig geworden. So kann sich eine philosophische Besinnung auf die wissenschaftliche Pragmatik der Informatik und die in ihr sich aufdrängenden Fundamentalfragen nicht zuletzt dadurch als fruchtbar erweisen, dass das in ihr wirksame aber eigens nicht thematisierte Welt- und Menschenverständnis expliziert und einer prüfenden Untersuchung unterzogen wird. Der philosophische Fragehorizont könnte seinerseits infolge einer solchen Eingelassenheit in die Besonderheiten einer informatorisch perspektivierten Weltwahrnehmung interdisziplinär angereichert werden.

Die Weltdeutung der Informatik als philosophische Frage

Zusammen mit anderen wissenschaftlichen Disziplinen wie Kybernetik, Gentechnik, Robotik, Nanotechnik und Bionik und verquickt mit ihnen stellt die Informatik einen der Gipfelpunkte der modernen Wissenschaft und Technik dar. Da aber der modernen Technik eine spezifische Weltdeutung zugrunde liegt, die sie geschichtlich von anderen möglichen Weisen der Weltvergewisserung unterscheidet, stellt sich die philosophische Frage nach den untersten, auch die Operationen der Informatik leitenden Prinzipien dieser Weise der Weltwahrnehmung. Eine solche Reflexion lässt sich auf eine aufschlussreiche Weise anhand der Erörterung jener Begrifflichkeiten durchführen, mit denen die Informatik stets operiert und die zugleich die zentralen Probleme der philosophischen Denktradition darstellen. Es handelt sich um das Problem der Objektbestimmung, das systematisch mit dem der Subjektbestimmung und der Objektivitätsdeutung einhergeht.

Es ist kein Zufall, sondern zeugt von der Zentralität dieses Problemkomplexes, dass fast die gesamte philosophische Technikkritik trotz der Diversität der Denkrichtungen und Thematisierungskontexte sich an der genannten Problematik und der Analyse der Bestimmung der Subjekt-Objekt-Beziehung durch das technisch geformte Bewusstsein orientiert. So wäre es ratsam, aufgrund der Präsenz dieser Thematik auch in den Bereichen wissenschaftlicher Tätigkeit sich mit dem Begriff des Objekts näher zu befassen und vor allem jene substantialistische Objekt-Auffassung kritisch zu hinterfragen, die zu theoretisch bedenklichen und praktisch gefährlichen Wahrheitskonzepten geführt haben.

Die philosophische Reflexion auf den Begriff des Objekts, der in einer einigermaßen äquivalenten Bedeutung bei Aristoteles im spezifisch griechisch-aristotelischen Sinne dieses

[1] G. Ch. Lichtenberg: Aphorismen, ausgewählt und eingeleitet von Friedrich Sengle, Stuttgart 2002, S. 41.

Begriffes Verwendung findet[2], ist überwiegend durch die ontologischen und epistemologischen Überlegungen veranlasst worden, die je nach der philosophischen Gesamtausrichtung in die Aufstellung diverser Objekttheorien einmünden. So hat die ontologische Fundmentalfrage der Philosophiegeschichte nach dem Existenzstatus der Außenwelt und die Weise ihrer Bestimmung in unterschiedlichen Denkrichtungen wie etwa in naivem, kritischem und transzendentalem Realismus, subjektivem und objektivem Idealismus sowie Phänomenalismus und Solipsismus zu entsprechend unterschiedlichen Objektkonzepten geführt. Demzufolge lässt sich eine angemessene Behandlung des Objektbegriffes nur im Kontext eines philosophischen und kulturellen Gesamtzusammenhangs und unter Berücksichtigung seiner systematischen Stellung im jeweiligen Weltbild vornehmen, das eine ihm adäquate Objektauffassung hervorruft.[3] Die Beschäftigung mit der objekttheoretischen Problematik vertieft sich somit zur Besinnung auf die Besonderheiten einer bestimmten philosophischen, ja kulturellen Weise der Weltapperzeption. So erfordert eine Auseinandersetzung mit dem Objektbegriff, so wie er in der europäischen Philosophietradition anzutreffen ist, die Explikation der spezifischen Denkvoraussetzungen, unter denen er entwickelt worden ist.[4]

Dualistisches Weltbild in der abendländischen Philosophie

Anders als die außereuropäischen Philosophietraditionen lässt sich die abendländische Philosophiegeschichte vor allem seit ihrer neuzeitlichen Phase in einer bestimmten Deutungsperspektive als die Geschichte eines dualistischen Denkens und die der Überwindungsversuche dieses Dualismus nachvollziehen.[5]

Bereits mit Platons Philosophie wird eine Weltdeutung eingeleitet, welche im Rahmen eines metaphysischen Ideen-Realismus dem Denken des Menschen die Ideen als ewig bestehende substantielle Wesenheiten zum Erkennen entgegenstellt. Diese Dualität von Denken und Sein, von Logik und Ontologie wird allerdings mittels der metaphysischen Annahme von der Strukturhomologie von Denken und Sein überbrückt, so dass aufgrund dieser grundlegenden Adäquation von Denk- und Seinsstrukturen die Erkenntnismöglichkeit der Ideen als der wesentlich Seienden verbürgt wird. Bedient man sich der Vernunft als des angemessenen Instruments der doxa-transzendenten Wesenserkenntnis, dann beansprucht eine solche Wesenserkenntnis den Status der Wahrheit, und zwar einer Wahrheit, der aufgrund ihres überzeitlichen Status durch substantiellen Wesensbezug die Momente der Notwendigkeit und der Allgemeingültigkeit - und das heißt unbedingte Geltung - zukommen.

So resultiert aus der essentialistischen Annahme der Ideen als der an sich seienden Objekte der Erkenntnis jener konsequenzreiche Objektivitäts- und Universalitätsanspruch, der seither

[2] Zur Begriffsgeschichte und dem Bedeutungswandel des Objektbegriffes vgl. Historisches Wörterbuch der Philosophie, hrsg. von Joachim Ritter und Karlfried Gründer, Bd. 6, S. 1026-1053.

[3] Kants kritizistische Suspendierung der Ontologie als leitender Grundlegungswissenschaft durch die Erkenntnistheorie, welche die metaphysische Ob-Frage nach dem Sein des Seienden durch die Wie-Frage der Erkennbarkeit der Gegenstände ersetzt und damit den metaphysischen Objektbegriff transzendentalphilosophisch revidiert, wäre aufgrund ihrer exzeptionellen Thematisierung dieser Problematik für die Erhellung dieses Sachverhaltes besonders instruktiv.

[4] n einer interkulturellen Reflexionsperspektive wäre die Frage aufschlussreich, warum das moderne technische Bewusstsein in seiner paradigmatischen Gestalt und mit seinen ambivalenten Folgen ausgerechnet im Westen, nicht aber im Osten entstehen konnte. Die kulturgeschichtlichen und denkspezifischen Voraussetzungen dafür aufzudecken, wäre die Aufgabe einer umfassenden, interkulturell sensibilisierten Philosophie der Technik.

[5] Der spätestens seit Platon zu verzeichnende, in der Neuzeit sich verschärfende metaphysische Dualismus in abendländischer Philosophietradition wurde vor allem durch Denkbewegungen wie philosophische Mystik, romantischen Holismus und spekulativen Idealismus zu überwinden versucht.

als das Grundmerkmal einer letztbegründeten Wahrheit beibehalten und in diversen Philosophiekontexten als das Attribut philosophischer Glaubwürdigkeit verteidigt wurde. Es sei angemerkt, dass der essentialistischen Welt- und Wahrheitsauffassung dieser Art schon früh philosophisch widersprochen wurde. Die pyrrhonische Skepsis lässt sich als antikes Projekt zur Entdogmatisierung der Wahrheitsfrage beschreiben, die durch die Unterscheidung von dem Wesen des Seienden, das sich der Erkenntnis entzieht, und der Erscheinung desselben als einem subjektiven Erlebnis nicht nur die antike Vorlage für Kantische Unterscheidung von Erscheinung und Ding an sich geliefert hat, sondern auch auf eine originäre Weise die Konzipierungsmöglichkeit einer relativistisch gedachten und doch verbindlichen Wahrheit in Erwägung gezogen hat.[6]

Das Platonische Seinsverständnis samt seinen objektmetaphysischen und wahrheitstheoretischen Implikationen wirkt strukturell in verneuzeitlichter Version in Descartes Denken fort. Im subjektzentrischen Unterschied zur antiken Philosophie und schärfer als je zuvor polarisiert Descartes im Horizont seiner Zentralisierung des Subjekts zum Prinzip der philosophischen Weltauslegung das Verhältnis des Menschen zur Welt. Dem als denkendes Ich substanzmetaphysisch konzipierten weltlosen Subjekt wird nachträglich eine als ausgedehnte Substanz gedachte Welt entgegengestellt, die für es den Status eines in seinen wesentlichen Strukturen mathematisch-adäquat erfassbaren Objekts einnimmt. Das Mensch-Welt-Verhältnis mutiert neuzeitlich zu einer substanzdualistischen Subjekt-Objekt-Relation.

Die Trennung des Subjekts von der Welt und die Statisierung ihres Verhältnisses im mecha nischen Modus einer starren Entgegensetzung gehören zu dieser Denkungsart als unvermeidliche Konsequenzen. Dem unbeseelten, mathematisch abstrahierten Weltobjekt steht das akosmische einsame Subjekt bezugslos gegenüber. Ausgehend von der Annahme von der Kommensurabilität der Bedingungen des Denkens mit denen des Seins hält dieses Denken an einer adäquaten Erkenntnismöglichkeit des Objekts durch das Subjekt gemäß der exaktwissenschaftlichen Wahrheitskriterien von Klarheit und Deutlichkeit fest. In einer so gewonnenen Erkenntnis wird diesem Denken gemäß das Objekt in adäquater Weise zugänglich, weshalb diese Erkenntnis Universalität und Objektivität beanspruchen kann.[7]

[6] So erklärt Sextus Empiricus: „Wir sagen nun, das Kriterium der skeptischen Schule sei das Erscheinende, wobei wir dem Sinne nach die Vorstellung so nennen; denn da sie in einem Erleiden und einem unwillkürlichen Erlebnis liegt, ist sie fraglos. Deshalb wird niemand vielleicht zweifeln, ob der zugrundeliegende Gegenstand so oder so erscheint. Ob er dagegen so ist, wie er erscheint, wird infrage gestellt." Sextus Empiricus: Grundriss der pyrrhonischen Skepsis, Einleitung und Übersetzung von Malte Hossenfelder, Frankfurt am Main 1968, S. 99.

[7] Bereits Goethe kritisiert diese szientistisch orientierte Wahrheits- und Objektivitätskonzeption, in der die Wahrheit durch jene, dem Bereich der naturwissenschaftlichen Rationalität zugehörigen Kriterien der Notwendigkeit und Allgemeingültigkeit qualifiziert und die mathematische Exaktheit zum Bestimmungskriterium jeglicher Wahrheitsfindung schlechthin totalisiert wird. So fragt Goethe in destruierender Herausstellung der exaktwissenschaftlichen Rationalität als unbewusster Emotionalität: " Was ist an der Mathematik exakt als die Exaktheit? Und diese, ist sie nicht eine Folge des inneren Wahrheitsgefühls?" Goethe: Maximen und Reflexionen, eingeleitet von Max Hecker, Frankfurt am Main 1976, Nr. 607.
Gepaart mit dieser Kritik bezweifelt Goethe die erkenntnisförderliche Potenz und die Zuverlässigkeit der exaktwissenschaftlichen Genauigkeit, wie sie in der von Descartes zu Wahrheitskriterien stilisierten Klarheit und Deutlichkeit akzentuiert wird: „Wir würden gar vieles besser kennen, wenn wir es nicht zu genau erkennen wollten." Ebd., Nr. 501.
Hieraus wäre Beachtliches zur Kritik jener Doktrin von der sogenannten Sachlichkeit zu gewinnen, die zu Irrlehren wie der von der Wertfreiheit der Wissenschaft oder von der politisch-gesellschaftlichen Neutralität wissenschaftlicher Grundlagenforschung verleitet hat.

Nietzsches Theorem von der Unmöglichkeit einer objektiven Wesenserkenntnis

Entscheidende philosophische Kritik erfährt dieses platonisch-cartesische Denkparadigma in Nietzsches Philosophie, der aufgrund der radikalen Zurückweisung desselben nicht nur die damit systematisch verbundene metaphysische Objekt- und Objektivitätsauffassung, sondern auch die gesamten Trägerinstanzen jener dualistischen Denkungsart wie Substanz, Subjekt, Vernunft und Logik kritisch-genealogisch depotenziert.

Verworfen wird vor allem die metaphysische Grundvoraussetzung von der wesensmäßigen Kompatibilität von Subjekt und Objekt, aufgrund derer das Objekterkenntnis unbedingte Geltung beanspruchen konnte.

Obwohl vor Nietzsche Kant durch seine kritische Erkenntnisanalyse die Unmöglichkeit einer derartigen Wesenserkenntnis begründet und die Erkenntnis auf den Bereich der Erscheinungen beschränkt hatte, hielt er jedoch seinem Transzendentalismus und Apriorismus zufolge an der Möglichkeit der Formulierung allgemeingültiger Sätze fest, die als geregelte Einheiten aus den synthetischen Leistungen des Bewusstseins resultieren. Durch seine Ablehnung des Kantischen Apriorismus bzw. durch den Erweis des Kontingenzcharakters des Apriorischen dekomponiert Nietzsche auch Kantisches Denken als neuzeitliche Metamorphose jenes traditionellen Denkparadigmas.[8]

Die Grundlage einer solchen Destruktionsarbeit bei Nietzsche ist seine Diagnose des reduktionistischen Charakters aller Erkenntnis, die stets ein Zurückbeziehen des Unbekannten auf das Bekannte, ein „sich-irgendwozu-in-Bedingung-setzen"[9] ist. Dieser erkenntniskritischen Einsicht entstammt Nietzsches Theorem von der Unmöglichkeit einer objektiven Wesenserkenntnis, die in der Lage wäre, die Doktrin von der unbedigten Wahrheitsgeltung epistemisch zu fundieren, denn „zwischen zwei absolut verschiedenen Sphären wie zwischen Subjekt und Objekt gibt es keine Causalität, keine Richtigkeit, keinen Ausdruck".[10] Indem Nietzsche auf diese Weise die traditionelle Adäquationstheorie der Wahrheit entkräftet und die Konditionalität und Konstruktionalität als die der Erkenntnis immanenten Restriktionsmomente exponiert, antiquiert er jede absolutistische Rede von einer endgültigen Wahrheitserfassung und entlarvt eine so totalitär gefasste Wahrheit als eine naive, seine Lokalität und Partialität vergessene Meinung: „das Wesen eines Dings ist auch nur eine *Meinung* über das „Ding". Oder vielmehr: das „*es gilt*" ist das eigentliche „*das ist*", das einzige `das ist´."[11]

[8] In unmissverständlicher Deutlichkeit distanziert sich Nietzsche mit Blick auf Humes Kausalitätskonzept vom Kantischen Apriorismus: "Die bestgeglaubten a priorischen „Wahrheiten" sind für mich - *Annahmen bis auf weiteres*, z. B. das Gesetz der Kausalität, sehr gut eingeübte Gewöhnungen des Glaubens, so einverleibt, dass *nicht daran* glauben das Geschlecht zugrunde richten würde. Aber sind es deswegen Wahrheiten? Welcher Schluss! Als ob die Wahrheit damit bewiesen würde, dass der Mensch bestehen bleibt!" Nietzsche: Der Wille zur Macht, 13. Aufl., Stuttgart 1996, S. 344.

[9] Ebd., S. 380.

[10] Nietzsche: Über Wahrheit und Lüge im außermoralischen Sinne, in: Kritische Studienausgabe (KSA) in 15 Bänden, hrsg. von G. Colli und M. Montinari, Bd. I, München 1988, S. 884.

[11] Der Wille zur Macht, S. 381. Dass die Verwechslung des „es gilt „ mit „das ist" die Verwechslung der Wahrheit als Konvention mit der als substantieller unbedingter Wesenheit bedeutet, deren epistemischer Besitzanspruch zu einer fundamentalistischen Gesinnung und destruktiven Praxis führt, erläutert Nietzsche anhand einer Analyse des Phänomens der Überzeugung und ihrer bedenklichen Voraussetzungen: „Überzeugung ist der Glaube, in irgend einem Punkt der Erkenntnis im Besitze der unbedingten Wahrheit zu sein. Diese Glaube setzt also voraus, dass es unbedingte Wahrheiten gebe; ebenfalls, dass jene vollkommenen Methoden gefunden seien, um zu ihnen zu gelangen; endlich, dass

Mit dieser Reduktion der metaphysischen Wahrheiten auf die Ebene hermeneutischer Interpretationen setzt Nietzsche nicht nur die wirkungsmächtige Platonische ontologische Hierarchie von scheinbarer und wahrer Welt und die ihr entsprechende Erkenntnispyramide von doxa und episteme außer Geltung, er leitet zugleich jene Hermeneutisierung der philosophischen Wahrheitsfrage ein, die von nun an die Dimensionen unaufhebbarer Bedingtheit, Endlichkeit und Horizonthaftigkeit als die konstitutiven Determinanten jeglicher Weltauslegung ernst nimmt und damit die dogmatisch-absolutistischen Wahrheitsbehauptungen auf ihre vielfache Sedimentiertheit hin befragt und dekonstruiert.

Überwindung der traditionellen Subjekt-Objekt-Dichotomie durch Heidegger

Durch Nietzsches essentialismuskritische Revision des Objekt- und Objektivitätsbegriffes inspiriert, gründet Heidegger seinen philosophisch-hermeneutischen Ansatz auf die fundamentalontologische Hinterfragung des herkömmlichen Subjekt-Objekt-Schemas, um es als eine abstraktive Auslegung des vorgängigen In-der-Welt-seins des Menschen zu interpretieren. So bezeichnet Heidegger die gesamte metaphysische Philosophietradition als eine solche, die das Sein im bedenklichen Rahmen einer Ontologie der Vorhandenheit stets zum Seienden vergegenständlicht und es dadurch vergessen hat. Flagrantes Beispiel einer solchen verobjektivierenden Denkungsart sei nach Heidegger bei Descartes anzutreffen, dessen Substanzdualismus in besonderer Weise die Trennung des Subjekts von Objekt kategorial fixiert und ihr Verhältnis als einen technisch-herrschaftlichen Okkupationsakt des Objekts durch das Subjekt organisiert.[12] Theoretisch unerklärlich bleibt zudem in einem solchen ontologischen Dualismus, wie eine Überbrückung dieser fundamental voneinander getrennten Substanzsphären überhaupt möglich sein soll. Hermeneutisch gewendet, hieße dies die Unmöglichkeit, ein verstehender Zugang zu dem radikal Differenten im Horizont einer solchen Konzeption herzustellen. So verwickelt sich das dualistische Denkparadigma in theoretisch unlösbare und praktisch problematische Schwierigkeiten, deren Behebung nur durch einen grundsätzlichen Paradigmenwechsel gelingen kann, in dem das Subjekt-Objekt-Schema aufgegeben und das Verhältnis des Menschen zur Welt anders gedacht wird.

Husserls bewusstseinstheoretischer Grundgedanke von der Intentionalität des Bewusstseins aufnehmend, die das Bewusstsein stets als Bewusstsein von etwas und somit in einem primordial und originär gegebenen Weltbezug konzipiert, leitet Heidegger seine hermeneutische Überwindung der traditionellen Subjekt-Objekt-Dichotomie ein. Eine solche Dichotomie ist nach Heidegger eine abkünftige Differenzierung, die nur vorgenommen werden kann, wenn zuvor und vorgängig eine dieser Unterscheidung vorausliegende Weltvertrautheit des Menschen gegeben ist: „Im Sichrichten auf...und Erfassen geht das Dasein nicht etwa erst aus seiner Innensphäre hinaus, in die es zunächst verkapselt ist, sondern es ist seiner primären Seinsart nach immer schon „draußen“ bei einem begegnenden Seienden der je schon entdeckten Welt.“[13]

In dezidierter Abhebung von jenem cartesisch-statischen Welt- und Menschenbild, das den Menschen abstrakt als weltabstinentes Subjekt konstruiert, bestimmt Heidegger die Welthaftigkeit als konstitutive Seinsart des Daseins, die allem prädikativen Denken vorausgeht und dieses erst ermöglicht.

Dieser existenzialanalytischen Suspendierung des traditionellen Objektverständnisses entspricht systematisch die Preisgabe jener unkritischen Wahrheitsauffassung, in der Wahrheit puristisch und objektivistisch als eine subjektiv unmodifizierte Objektivität begriffen wurde.

Der hermeneutischen Wende zufolge ist Erkenntnis welcher Art auch immer nur aufgrund der originären Weltlichkeit und Weltbezüglichkeit des Menschen möglich. Mit anderen Worten ist das Verstehen von etwas allein deshalb möglich, weil die Welt dem Menschen immer schon erschlossen ist, so dass das Verstehen durch das Bereits-verstanden-haben ermöglicht wird. Auf diese dem Verstehen intrinsisch zukommende Struktur der Voraussetzungshaftigkeit hinweisend, heißt es bei Heidegger prägnant: „Auslegung ist nie ein voraussetzungsloses Erfassen eines Vorgegebenen.“[14]

Das Verstehen, von dem auch wissenschaftliche Erkenntnis ein Spezialmodus ist, geschieht somit in einem vorverstandenen, primordial erschlossenen Sinnzusammenhang, weshalb jedem Verstehen der Charakter unabstreifbarer „Vorstrukturiertheit“ innewohnt.[15] Die „Vor-Struktur“ des Verstehens zeigt als das, worin die Welt dem Menschen originär zugänglich ist, die Möglichkeit, aber auch die Grenze des Verstehens an, indem sie jede Erkenntnis als eine im spezifisch hermeneutischen Sinne voreingenommene Auslegung vorprägt, die das Unbekannte im Modus des bereits Bekannten erkennt und dadurch ihre zwar verschiebbaren aber nie aufhebbaren Geltungsgrenzen erhält. Dieser durch die Vorstrukturiertheit angezeigte restriktive Effekt, welcher der Erkenntnis die Konstitutionsmerkmale der Fragmentarität und Perspektivität aufprägt, weist die These von einem anfänglichen totalen Unverständnis im Sinne eines leeren und weltabgeschiedenen Bewusstseins ebenso ab wie die Behauptung eines endgültigen Finalverständnisses.

Das Konzept eines offenen Subjekts

Im Kontext einer solchen Erkenntnisauffassung, welche die Endlichkeit, Geschichtlichkeit und Vorläufigkeit als immanente Ingredienzien aller Erkenntnis exponiert, ließe sich nicht mehr eine geltungs-totalitaristische Wahrheitstheorie oder eine puristische Objektivitäts-

---

Jeder, der Überzeugungen habe, sich dieser vollkommenen Methoden bediene. Alle drei Aufstellungen beweisen sofort, dass der Mensch der Überzeugungen nicht der Mensch des wissenschaftlichen Denkens ist.“ KSA 2, S. 356. Es lässt sich wie eine Warnung vor essentialistischen Wahrheitstheorien vernehmen, wenn Nietzsche auf die verheerenden Folgen derartiger exklusivistischer Wahrheitsbehauptungen in der Geschichte aufmerksam macht, die zu ideologischen Glaubensartikeln reifiziert werden: „Es ist nicht der Kampf der Meinungen, welcher die Geschichte so gewalttätig gemacht hat, sondern der Kampf des Glaubens an die Meinungen.“ Ebd, S. 356.

[12] So schreibt Heidegger im Rahmen seiner Beschreibung der neuzeitlich-cartesischer Weltauslegung, in der das Seiende im Modus des Vorgestellten vergegenständlicht wird: „Diese Vergegenständlichung des Seienden vollzieht sich in einem Vor-stellen, das darauf zielt, jegliches Seiende so vor sich zu bringen, dass der rechnende Mensch des Seienden sicher und d.h. gewiss sein kann...Erstmals wird das Seiende als Gegenständlichkeit des Vorstellens und die Wahrheit als Gewissheit des Vorstellens in der Metaphysik des Descartes bestimmt.“ Heidegger: Die Zeit des Weltbildes, in: Holzwege, Frankfurt am Main 1950, 69-104, S. 80.

[13] Heidegger: Sein und Zeit, 16. Aufl., Tübingen 1986, S. 62.

[14] Ebd., S. 150.

[15] In Heideggers eigener Diktion heißt es diesbezüglich: „Das Verstehen betrifft als die Erschlossenheit des Da immer das Ganze des In-der Welt-seins. In jedem Verstehen von Welt ist Existenz mitverstanden und umgekehrt. Alle Auslegung bewegt sich ferner in der gekennzeichneten Vor-Struktur. Alle Auslegung, die Verständnis beistellen soll, muss schon das Auszulegende verstanden haben.“ Ebd. S. 152.
Es gilt hervorzuheben, dass Heideggers Theorem von der Vor-Struktur des Verstehens, das in Gadamers Hermeneutik als das Konzept der Vorurteilhaftigkeit des Verstehens rezipiert wird, einen absolutismus- und totalitarismuskritischen Theoriebestandteil seines Denkens darstellt, der im Geiste der Erarbeitung einer offenen und gewaltfreien Hermeneutik fruchtbar gemacht werden könnte.

doktrin aufrechterhalten. Mit der hermeneutischen Verabschiedung des metaphysisch-essentialistischen Objektbegriffes und der ihm korrespondierenden subjekt-und wahrheitsdogmatischen Voraussetzungen eröffnet sich ein Denkraum, in dem ideologiekritische Potentiale aufdeckbar sind, die zur Entwicklung einer pluralitätsbewussten Intersubjektivitätskonzeption innerhalb einer vielfaltwahrenden Geisteshaltung als des dringenden Erfordernisses einer polyphon geformten Lebenswirklichkeit durchaus entfaltet werden könnten.

Das distinktive Merkmal einer solchen Geisteshaltung besteht in ihrem andersartigen Verständnis des Subjekts, das dieses nicht mehr im Sinne einer überweltlichen Entität, sondern als ein dynamisches Werdephänomen begreift, das erst im konkreten intersubjektiven Beziehungsgeflecht entsteht und wandelt. Als philosophische Aufgabe gälte es deshalb, jenseits der einseitigen Extrempositionen subjekt-absolutistischer und subjekt-dekonstruktivistischer Art ein differenzbewusstes und zugleich kommunikationsfähiges Subjektkonzept zu erarbeiten. In Abhebung von dualistischen Konzepten, die das Subjekt-Welt-Relation konstrukthaft polarisieren, versucht eine offene Subjektauffassung, die Beziehung zwischen Subjekt und Welt konzeptionell zu symmetrisieren. Sie geht deshalb vom Primat der Intersubjektivität vor der Subjektivität aus, der es erlaubt, die insulare Hermetik des einsamen Subjekts aufzubrechen und es auf diese Weise zu sozialisieren. Mit der Aufwertung der sozialen Welt zur notwendigen Bedingung der Subjektivität und Positionierung derselben innerhalb der konkreten Interaktionsbereiche wird sie in ihrem autoritären Verhältnis zur Welt entschärft, indem der Welt zugleich der Charakter des dem Subjekt subordinierten Objektseins genommen wird. Praktisch würde dies den Versuch bedeuten, den weltverlorenen Menschen in einen gewaltfreien Bezug zur Welt zu setzen und sein Verhältnis zu ihr elementar zu befrieden.

Das Objekt der Informatik als Gegenstand der Softwarebeschreibung

Bettina Müller

München
mueller_bettina@gmx.de

Das Problem des Objektbegriffs in der Informatik

Ziel dieses Beitrags ist es, auf der Grundlage des semiotischen Ansatzes von Charles S. Peirce den Objektbegriff in der Informatik näher zu bestimmen. Der Terminus „Objektorientierung" ist bereits vor über 20 Jahren entstanden und innerhalb der letzten 10 Jahre zu dem vorherrschenden Paradigma der Informatik - besonders der Softwareentwicklung - avanciert.[1] In den letzten Jahren haben sich in der Informatik zahlreiche Theorien zur objektorientierten Softwareentwicklung heraus gebildet, aber mit nahezu jeder dieser Theorien wird eine weitere Perspektive auf die Objekte entworfen. Zwar herrscht in der Informatik die einstimmige Meinung, dass die Objekte der Programmierung einem Bündel von Daten und Operationen entsprechen. Es steht auch außer Zweifel, wie diese in Programmcode umzusetzen sind. Unklar ist jedoch bislang, wie die Programmeinheiten mit unserer alltäglichen Vorstellung von Objekten zu vereinbaren sind. Es besteht lediglich die vage Idee, die Einheiten mit unserer Alltagsvorstellung von Objekten in Beziehung zu setzen. Doch welche Anknüpfungspunkte bieten die realen Objekte für die Softwareentwicklung? Inwiefern können die realen, gegebenen Objekte in die Konstruktion der Softwareentwicklung integriert werden? Lassen sich die gegebenen Entitäten ohne weiteres mit den konstruierten Entitäten der Informatik vergleichen? Und welcher Vergleich ist hier überhaupt sinnvoll? Auf diese Fragen findet man in der Informatik keine klaren Antworten. Hier zeigt sich, dass der Objektbegriff in der Informatik in theoretischer Hinsicht noch weitgehend unterbestimmt ist.

Die objektorientierten Vorgehensmodelle zur Softwareentwicklung legen verschiedene Objektkonzeptionen zugrunde. Besonders in der objektorientierten Analyse finden sich ganz unterschiedliche Strategien, um die Softwareeinheiten mit unserem alltäglichen Objektverständnis in Verbindung zu bringen. Hier bezieht man sich meist auf Objekte des Problembereichs bzw. der realen Welt, wie beispielsweise physische Gegenstände, Personen, Orte, Rollen, Interaktionen, Dienste, Ereignisse, Strukturen, Konzepte, usw.[2] Wie an diese Objekte heranzutreten ist oder welche Objekte für die Softwareentwicklung von Bedeutung sind, wird mit jeder Strategie anders gelöst. Unterschiedliche Vorstellungen über die Objekte der realen Welt prägen die einzelnen Vorgehensweisen.

Einer der populärsten Ansätze zur Objektfindung ist der von I. Jacobson[3]. Er empfiehlt zunächst typische und besondere Anwendungsfälle[4] detailliert darzulegen, um die gewünschte

[1] Vgl. z.B. [Quibeldey-Cirkel 94] S. 2 f., 140
[2] Vgl. [Booch 94] S. 155 f.
[3] Vgl. [Jacobson et al. 92]

Funktionalität des zu entwickelnden Systems zu beschreiben. Danach können durch Analyse der Anwendungsfälle die ersten Objekte und ihre Beziehungen zueinander konstruiert werden. Hier wird den Objekten also eine Beschreibung der Software als dynamisches System zugrunde gelegt.

J. Rumbaugh et al.[5] schlagen als ersten Analyseschritt vor, die Substantive, Adjektive und Verben im Anforderungs- und Spezifikationstext der zukünftigen Software zu betrachten.[6] Dabei soll geprüft werden, ob sie sich für Objekte, Objekteigenschaften oder Objektbeziehungen eignen. Damit ergibt sich ein erstes statisches Objektmodell, dem eine dynamische Modellierung der Zustände und Ereignisse und eine funktionale Modellierung folgen. Diese Vorgehensweise orientiert sich streng an den im Text vorgegebenen Bezeichnungen für Objekte, Objekteigenschaften und Objektbeziehungen.

Betrachtet man diese und andere Verfahren zur objektorientierten Analyse, wird schnell klar, dass sie ganz unterschiedliche Aspekte der Problemstellung bzw. der realen Welt hervorheben. Anknüpfungspunkt für die Objekte der Informatik bilden hier verschiedene Zugangsweisen zu einem Realitätsausschnitt. Meist wird dabei auf das Alltagsverständnis zurückgegriffen und die Beziehung von Objekt, Subjekt und Wirklichkeit wird nicht weiter thematisiert. Eine kritische Analyse des Objektbegriffs wurde in der Informatik bislang kaum unternommen. Mit der „inflationären Ausweitung des Objektbegriffs"[7] innerhalb der letzten Jahre ist diese umso dringender geboten. Für die Informatik ist es wichtig zu wissen, welche Vorstellungen sie zugrunde legt und ob diese für die Zwecke der Informatik überhaupt dienlich sind.

Beispielsweise geht es in der Analysephase „... idealerweise darum, daß EntwicklerInnen und BenutzerInnen eines Softwaresystems sich gemeinsam eine Vorstellung davon bilden, was eigentlich entwickelt werden soll, was das *Problem* ist und wie eine mögliche informatische Lösung dafür aussehen könnte".[8] Schinzel bemängelt, dass dieses Idealbild in der Praxis kaum zu finden sei. Im Gegenteil - so Schinzel - werde die Beeinträchtigung der Qualität von Software und der Produktivität laut einer Studie des Forschungsteams um Bill Curtis besonders durch drei Faktoren geprägt:[9]

- zu geringe und zu wenig verbreitete Kenntnisse der EntwicklerInnen über das Anwendungsgebiet
- sich verändernde und widersprüchliche Anforderungen an das Softwaredesign sowie

[4] Anwendungsfälle beschreiben die Interaktionen zwischen dem System und den Aktoren. Aktoren sind beispielsweise Benutzer, andere Systeme oder Geräte, die von außen mit dem System Informationen austauschen.
[5] Vgl. [Rumbaugh et al. 93]
[6] Diese Methode der Textanalyse geht auf Abbot zurück: Abbot R.J.: Program Design by Informal English Descriptions, Communications oft the ACM 26 (11), S. 883-894 (vgl. [Rundshagen, Schader 94] S. 48)
[7] [Hesse, von Braun 01]
[8] [Schinzel et al. 99] S. 12
[9] Vgl. ebenda S. 13

- Engpässe und Zusammenbrüche in der Kommunikation und Zusammenarbeit.

Dieser Umstand wird auch von den meisten Theorien zur objektorientierten Analyse kaum berücksichtigt. Sie beschreiben hauptsächlich die methodische Lösung einer vorgegebenen Aufgabenstellung.[10] So herrscht in der Informatik oftmals die Doktrin „das Tun (und auch die Aufgabe) des Informatikers [beginnt] dort, wo eine eindeutige Aufgabenstellung oder Problemformulierung vorliegt."[11] Häufig geht es nur um die objektorientierte Beschreibung von bereits Analysierten.[12]

So ist es nicht verwunderlich, wenn die Problemstellung bzw. ein Realitätsausschnitt als vorgegebenes Original betrachtet wird, das es abzubilden gilt. Dementsprechend werden die Objekte der Informatik oftmals als Abbilder von realen Gegenständen verstanden. Diese Abbildungsperspektive geht meiner Meinung nach aus den naturalistischen Grundannahmen der Informatik hervor. Gegenstand naturwissenschaftlicher Untersuchungen ist etwas Vorgegebenes, das auf seine Gesetzmäßigkeiten hin untersucht wird. Die Entdeckung von Naturgesetzen beeinflusst (nach diesem Verständnis) nur unser Wissen über den Gegenstand, aber nicht den Gegenstand selbst. Dementsprechend wird in der Analysephase angenommen, dass der gegebene Sachverhalt oder Realitätsausschnitt ein unveränderbares Original darstellt, das objektorientiert modelliert werden soll. Modellierung wird hier im Sinne von Abbildung verstanden.

Sobald man jedoch aus der naturwissenschaftlichen Praxis des Beobachtens heraustritt und das eigene Tun sowie die Methoden reflektiert, stellt sich das Verhältnis zum Objekt anders – nämlich als Konstruktionsergebnis dar. In der Konstruktionsperspektive erscheint das Gegebene - in unserem Fall die Ausgangslage bzw. Problemsituation - nicht länger als unabänderlich, sondern vielmehr als etwas, das durch einen Prozess entstanden ist und sich auch weiterhin verändern kann. Gegenstand der objektorientierten Analyse ist hier nicht ein unveränderliches Original, das abgebildet werden soll, sondern eine verhandelbare Aufgabenstellung, die das Ziel und den Zweck der Softwareentwicklung festlegt.

In ihrer meinungsbildenden und handlungsleitenden Funktion sollten die Vorgehensmodelle über einen zweckdienlichen und theoretisch legitimierbaren Objektbegriff verfügen. Je größer das Softwareprojekt ist, umso wichtiger wird ein methodisches Vorgehen, das letztendlich zu einer qualitativ hochwertigen Software führen soll.[13] Um den Anspruch der systematischen Entwicklung von Software zu genügen, benötigt die Informatik in ihrer Funktion als Wissenschaft eine klare Objektkonzeption. Solange in der Informatik der Anspruch besteht,

[10] Vgl. [Rödiger 95] S. 212
[11] [Busse 99] S. 145
[12] Vgl. [Rödiger 95] S. 212
[13] Die methodischen Hilfsmittel zur Softwareentwicklung sind bis heute nicht ganz ausgereift. Besonders in sehr großen Softwaresystemen werden schwerwiegende Fehler oft zu spät entdeckt. (vgl. [Gumm, Sommer 00] S. 642) Dies ist besonders problematisch, wenn Fehlfunktionen gravierende Konsequenzen haben. Wie beispielsweise ein Zugunglück, das letztendlich durch eine fehlerhafte Software verursacht wurde.

die Programmeinheiten mit der „natürlichen" Weltsicht in Verbindung zu bringen, muss der Objektbegriff in der Informatik auch ein zweckmäßiges Verständnis der Realität umfassen.

Eine allgemeine Reflexion über die Wirklichkeit, die „realen" Objekte und ihre Beziehung zum Subjekt ist aber nicht Aufgabe der Informatik, sondern die der Philosophie. Im Gegensatz zu unserem Alltagsverständnis stellt die Philosophie Konzepte über die Objekte unserer Alltagserfahrung bereit. Besonders die Erkenntnistheorie bietet Reflexionen über die Objekte der realen Welt, wie sie zu erkennen und benennen sind.

Peirces Konzeption des Objektbegriffs

Im Folgenden soll eine Objektkonzeption vorgestellt werden, die eine passende Sichtweise von Realität aufzeigt und die dem „Objektfindungsprozess" in der Analyse zugrunde gelegt werden kann. Der semiotische Ansatz von Charles S. Peirce bietet einen geeigneten Rahmen, um unsere „natürliche" Weltsicht mit dem objektorientierten Ansatz der Informatik zu vereinbaren: Genau wie die Softwareobjekte sind die realen Objekte uns nicht unmittelbar gegeben, sondern Gegenstand eines Entwicklungsprozesses. In diesem Prozess können sie durch verschiedene Zeichen zum Ausdruck gebracht werden; beispielsweise durch die natürliche Sprache oder auch durch Programmcode. Vermittels Zeichen ist es möglich, etwas zu erkennen, darzustellen oder mitzuteilen. Da die Zeichentheorie nicht nur die sprachliche Repräsentation, sondern jegliche Repräsentationsart behandelt, eignet sie sich besonders gut als Grundlagenwissenschaft. Peirces Untersuchung der Zeichen bzw. der Repräsentation von Objekten hat die Semiotik nachhaltig geprägt. Heute gilt Peirce als der Hauptbegründer der modernen Semiotik.[14]

Nach Peirce ist ein Objekt grundsätzlich Gegenstand einer Zeichenrepräsentation. Sowohl die realen Objekte, als auch die softwaretechnischen Objekte sind für uns nur innerhalb einer triadischen Zeichenrelation von Zeichenträger, Objekt und Interpretant erfahrbar. Ein Zeichenträger bzw. eine materielle Erscheinungsform repräsentiert ein Objekt, wenn es vermittels eines Interpretanten als Zeichen für dieses Objekt gedeutet wird. Mit anderen Worten: Wenn wir ein Objekt erkennen oder darstellen möchten, ist dies uns nur vermittels Zeichen möglich. Wir haben keinen direkten Zugang zu einem Objekt. Erst wenn wir ein wahrnehmbares Etwas als Zeichen interpretieren, das für ein Objekt steht, wird ein Objekt für uns fassbar. Peirce zufolge gibt es weder ein Zeichen, das nichts darstellt, also für kein Objekt steht, noch Objekte, die nicht durch Zeichen repräsentierbar sind. Zeichen und Objekt treten immer gemeinsam auf, weil das Objekt nach Peirce Teil einer Zeichenrelation ist. Zeichen und Objekt sind jedoch weder willkürlich miteinander verknüpft, geschweige denn identisch. Vielmehr kann ein Objekt durch unterschiedliche Zeichen zum Ausdruck gebracht werden. (Genauer gesagt ist es der Zeichenträger, der dem Zeichen seine materielle Erscheinung gibt und damit das Objekt zum Ausdruck bringt.) Ein und dasselbe Objekt kann beispielsweise durch das gedruckte Wort „Hausdach", einer bestimmten Lautfolge, einem Ikon oder durch

[14] Vgl. [Oehler 93] S. 126

eine andere sinnlich wahrnehmbare Erscheinung zum Ausdruck gebracht werden. Aber auch ein Zeichenträger kann verschiedene Objekte repräsentieren. Beispielsweise kann das Ikon ein Hausdach oder einen Briefumschlag darstellen.

Doch wie kommt nun die Verbindung zwischen Objekt und Zeichenträger zustande? Die Verknüpfung der beiden Zeichenrelate Zeichenträger und Objekt geschieht durch den Interpretanten, dem dritten Relat der dreistelligen Zeichenrelation. Erst durch einen Interpretanten

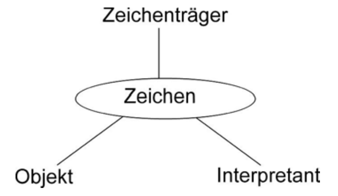


Abbildung 1: Die triadische Zeichenrelation

wird eine materielle Erscheinungsform, wie beispielsweise , zum Zeichen für ein Objekt. Je nach Interpretant steht als Zeichen für ein Hausdach aus der Vogelperspektive oder einen Briefumschlag.

Die Repräsentation von softwaretechnischen Objekten

Auch die softwaretechnischen Objekte der Informatik sind immer Gegenstand einer Zeichenrepräsentation:

1. Die Objekte werden durch verschiedene Zeichenträger repräsentiert. Schriftzeichen der natürlichen Sprache, der Programmiersprache und grafische Zeichen der Unified Modeling Language (UML) sind solche Zeichenträger und durch ihre jeweiligen materiellen Qualitäten wahrnehmbar.
2. Die reale Verbindung zwischen Zeichenträger und Objekt zeigt sich daran, dass sich das Objekt verändert, wenn der Zeichenträger verändert wird. Beispielsweise ändert sich tatsächlich das Objekt, wenn wir in einer UML-Grafik das Zeichen „rot“ ausradieren und stattdessen „grün“ schreiben. Fortan ist das repräsentierte Objekt nicht mehr rot, sondern grün.

   Dies setzt natürlich voraus, dass wir „rot“ bzw. „grün“ als die materielle Erscheinungsform eines Zeichen, d.h. als Zeichenträger interpretieren. Das Zeichen „rot“ bzw. „grün“ steht in einer UML-Grafik meist für das Attribut eines softwaretechnischen Objektes.[15]
3. UML-Grafiken besitzen nicht immer eine repräsentative Funktion. Ein Auftraggeber weiß meist nicht, was die grafischen Symbole bedeuten sollen. Erst, wenn man die Zeichen zu deuten weiß, können sie einem Objekt zugeordnet werden. Erst mit dem Interpretanten wird etwas zu einem Zeichen für ein Objekt. In der Softwareentwicklung hat der Entwickler mit verschiedenen Repräsentationsformen der Objekte zu tun. Kommuniziert er mit dem Kunden, dann muss er die natürliche Sprache benutzen. Wendet er sich an den Computer, hat er die Programmiersprache zu verwenden.

Spricht man in der Informatik von Objekten, so müssen auch hier Zeichenträger verwendet werden, um die Objekte zum Ausdruck zu bringen und erst durch einen entsprechenden Interpretanten wird eine materielle Erscheinungsform zu einem Zeichen für ein Objekt.

Der Objektfindungsprozess

Der Objektfindungsprozess lässt sich ebenfalls durch Peirces Objekt- bzw. Zeichenkonzeption veranschaulichen: Ziel der objektorientierten Analyse ist es, ein adäquates Objektmodell aufzustellen, das die gewünschte Software repräsentiert. Um dieses Ziel zu erreichen, wird die Software in der Analysephase auf zweierlei Weise beschrieben: in Absprache mit dem Kunden bzw. dem Anwender in Form der natürlichen Sprache und in Hinblick auf die Programmierung durch ein Objektmodell, das in Form von UML-Grafiken zum Ausdruck gebracht wird. Diese Softwarebeschreibungen repräsentieren in der Analysephase einen Sollzustand, den es im Laufe des Entwicklungsprozesses zu realisieren gilt. Dieser Sollzustand ist jedoch nicht feststehend, sondern verändert sich mit den Anforderungen an das Produkt. Die erfolgreich realisierte Software stellt sich dann in Form eines funktionsfähigen Programms dar, das alle gewünschten Anforderungen erfüllt.

Grundlage für den Entwurf eines Objektmodells ist die normative Softwarebeschreibung in Form der natürlichen Sprache. Doch wie kommt es nun zu den einzelnen Objekten in dem Modell? Ähnlich wie bei den Objekten der realen Welt liegt ihnen ein Erkenntnisprozess zugrunde, der in einer adäquaten Objektrepräsentation mündet. Wir erschließen uns ein reales Objekt hypothetisch, um unsere phänomenalen Erfahrungen zu erklären. Objekte entstehen durch Hypothesen, um dem Kohärenz zu verleihen, was sonst ein nicht verstehbares Chaos von Sinneseindrücken wäre. Die Hypothese von einem Objekt wird durch das Faktum, dass sie mit unserer phänomenalen Erfahrung übereinstimmt und diese erklärt, bestätigt.[16]

---

[15] Das Objekt der peirce'schen Zeichenrelation kann jedoch auch einem vollständigen softwaretechnischen Objekt entsprechen. Dies ist der Fall, wenn ein Interpretant die entsprechende Erscheinungsform in einer UML-Grafik (eine einspaltige Tabelle mit einem unterstrichenen Wort in der ersten Reihe) als ein Zeichen für ein softwaretechnisches Objekt zu deuten weiß.

[16] Durch schlussfolgerndes Denken wird eine kausale Ordnung postuliert, in der das Objekt die Wahrnehmung verursacht. Tatsächlich ist es aber so, dass der Begriff des Objektes erst durch einen hypothetischen Schluss entsteht. Die von uns gedachte Ursache der Wahrnehmung ist also das Produkt eines Schlussprozesses, der von der Sinneswahrnehmung ausgeht. (Allerdings ist die Vorstellung einer Sinneswahrnehmung auch wiederum das Produkt eines Schlussprozesses.)

Die softwaretechnischen Objekte werden zwar nicht herausgebildet, um unsere phänomenalen Erfahrungen zu erklären, aber auch sie dienen dazu, einen komplexen Sachverhalt zu ordnen. Ähnlich wie bei den realen Objekten, wird durch Hypothesenbildung der Versuch unternommen, etwas zu ordnen bzw. zu vereinheitlichen oder zu klassifizieren. In der objektorientierten Analyse entstehen die softwaretechnischen Objekte, indem Informationen und Operationen sinnvoll zusammengefasst werden. Ich schlage deshalb vor, in der objektorientierten Analyse erst die Informationen und Operationen zusammenzutragen und auf dieser Grundlage die Objekte zu bilden.[17]

Für die Herausbildung von Objekten sind hier nicht reale Objekte maßgebend, sondern lediglich die Informationen und Funktionen, des zukünftigen Softwaresystems. Die so entstandenen Entitäten können dann jedoch durchaus bis zu einem gewissen Grad mit bereits bekannten Objekten der realen Welt übereinstimmen. Nachdem das Modell vorliegt, mag es möglicherweise vorteilhaft sein, die Objekte darin mit bereist Bekanntem zu vergleichen. Dadurch wird es einfacher, die nun formale Softwarebeschreibung zu verstehen. Zum Objektbildungsprozess trägt diese Sichtweise allerdings nicht bei. Ziel der Objektbildung ist es, relevante Informationen und Operationen zusammenzufassen und nicht etwas Reales abzubilden.

Den Objekten liegt hier in erster Linie eine normative Softwarebeschreibung in natürlicher Sprache zugrunde. Diese Beschreibung umfasst auch reale Objekte. Ziel der Objektbildung ist es jedoch nicht, diese Objekte abzubilden. Vielmehr geht es darum, die zu verarbeitenden Informationen und die Operationen bzw. Funktionen der Software zu erkennen, diese sinnvoll zusammenzufassen und dann in einem Objektmodell darzustellen. Im weiteren Verlauf werden die softwaretechnischen Objekte zunehmend ausgearbeitet.

Ob das Objektmodell tatsächlich die gewünschte Software repräsentiert, zeigt sich vor allem dann, wenn diese Objekte in Programmcode übersetzt werden und das Programm wunschgemäß funktioniert. Oftmals stellt der Anwender dann fest, dass eine wesentliche Funktion der Software fehlt. Die softwaretechnischen Objekte müssen nun auf der Basis der neuen Anforderungsspezifikation korrigiert werden. Im Entwicklungsprozess sind die Objekte durch Prototyping einer ständigen Verifikation unterworfen. Sie sind keine starren Gebilde, sondern verändern sich mit den Anforderungen an die Software.

Zusammenfassend kann festgestellt werden, dass die objektorientierte Softwareentwicklung keine Abbildung von realen Objekten ist, sondern ein Prozess, in dem auf verschiedene Weisen dargestellt wird, was die Software genau leisten soll. Zuerst in natürlicher Sprache, dann in Form eines Objektmodells und schließlich in einer Programmiersprache.[18] Jede Repräsentationsform spricht andere Interpreten an. Die Interpreten der natürlichen Sprache sind Auftraggeber bzw. Anwender und Entwickler. Das Objektmodell wird von dem Entwickler interpretiert. Das Programm - dargestellt in einer bestimmten objektorientierten Programmiersprache - versteht der Entwickler zum einen als Repräsentation von bestimmten Objekten und Objekttypen und zum anderen aber auch wie der Rechner: als Repräsentation von gebündelten Daten und Rechenanweisungen. Ob das Ziel der Softwareentwicklung erreicht wurde und ob die einstmals entworfenen softwaretechnischen Objekte ihren Zweck erfüllt haben, zeigt sich letztendlich dann, wenn die Software den Anwender wunschgemäß unterstützt.

[17] Um bei größeren Softwareprojekten nicht den Überblick zu verlieren, müssten die relevanten Informationen und Operationen zunächst nach bestimmten Gesichtspunkten geordnet werden. Auf dieser Grundlage könnten sich dann die einzelnen Objekte konstituieren.

[18] Zu beachten ist, dass in einem iterativen Softwareentwicklungsprozess die einzelnen Entwicklungsphasen (Analyse, Design und Realisierung) nicht nur einmal, sondern mehrmals durchlaufen werden.

Literaturverzeichnis

[Booch 94] Grady Booch: ObjektOriented Analysis And Design with Applications, 2. Auflage, Redwood City, California: Benjamin-Cummings, 1994

[Busse 99] Johannes Busse: Attribution von Verantwortung durch Metaphernanalyse, Dissertation, Universität Tübingen, Wilhelm-Schickard-Institut, 1999, Online im Internet: URL: http://www-pu.informatik.uni-tuebingen.de/users/busse/diss/dissA4.ps (Stand: 21.12.2002)

[Gumm, Sommer 00] Heinz-Peter Gumm, Manfred Sommer: Einführung in die Informatik, unter Mitw. von Bernhard Seeger und Wolfgang Hesse, 4. überarb. Auflage, München, Wien: Oldenbourg, 2000

[Hesse, von Braun 01] Wolfgang Hesse, Hubert von Braun: Wo kommen die Objekte her? Ontologisch-erkenntnistheoretische Zugänge zum Objektbegriff, Workshop "Erkenntnistheorie-Semiotik-Ontologie (ESO): Die Bedeutung philosophischer Disziplinen für die Softwaretechnik", in: K. Bauknecht, W. Brauer, Th. Mück: Informatik 2001 - Tagungsband der GI/OCG-Jahrestagung, 25. - 28. September 2001, Universität Wien, Bd. II, S. 776-781, books_372ocg.at, Bd. 157, Österr. Computer-Gesellschaft 2001

[Jacobson et al. 95] I. Jacobson, M. Christerson, P. Jonsson et al: Object-Oriented Software Engineering – A Use Case Driven Approach, Addison-Wesley, 1992

[Oehler 95] Klaus Oehler: Sachen und Zeichen, Zur Philosophie des Pragmatismus, Frankfurt am Main: Vittorio Klostermann, 1995

[Quibeldey-Cirkel 94] Klaus Quibeldey-Cirkel: Das Objekt-Paradigma in der Informatik, Stuttgart: Teubner, 1994

[Rödiger 95] Karl-Heinz Rödiger: Arbeitsanalyse und Software-Entwicklung, in: Friedrich J.: Informatik und Gesellschaft, Heidelberg; Berlin; Oxford: Spektrum Akademischer Verlag, 1995

[Rumbaugh et al. 93] J. Rumbaugh et al: Objektorientiertes Modellieren und Entwerfen, Hanser & Prentice-Hall, 1993

[Rundshagen, Schader 94] Michael Rundshagen, Martin Schader: Objektorientierte Systemanalyse, Heidelberg: Springer, 1994

[Schnizel et al. 99] Schinzel Britta, Parpart Nadja, Westermayer Til: Informatik und Geschlechterdifferenz, Tübinger Studientexte Informatik und Gesellschaft, Universität Tübingen, Wilhelm-Schickard-Institut für Informatik, 1999

Thesen zu Gegenstand, Zeichen, Objekt - und deren Bedeutung für die objektorientierte Analyse

Wolfgang Hesse[1] , Hubert v. Braun[2]

[1] FB Mathematik und Informatik, Univ. Marburg, hesse@informatik.uni-marburg.de
[2]ASG München, hubert.vbraun@uumail.de

Zusammenfassung: In diesem Beitrag beleuchten wir einige Begriffe, die für die in der Informatik seit ca. 1990 propagierte und praktizierte "objektorientierte Analyse und Modellierung" (OOAM) wichtig sind. Dazu gehören u.a. *Gegenstand, Zeichen, Bezeichner* und *Bezeichnetes, Konzept, Bezug* (referent), und *Objekt*. Während "Gegenstand" als unbestimmter "Universal-Designator" erkannt und eingestuft wird, kommt "Objekt" (wenn es nicht synonym zu Gegenstand verwendet wird) - je nach Kontext - eine spezifischere Bedeutung zu. Das gilt vor allem in der Informatik für den Kontext der "Objektorientierung". Aber auch hier findet man sehr unterschiedliche Begriffsverständnisse nebeneinander. Dem Anspruch der OOAM, (nicht-symbolische) "Realwelt" und (symbolische) "Software-Welt" miteinander zu verbinden, kann man gerecht werden, wenn man drei Aspekte von "Objekt" gemeinsam betrachtet: den universell-ontologischen (U-Aspekt), den konzeptionellen (C-Aspekt) und den Repräsentations- (R-) Aspekt.

Einleitung: Warum wir uns mit dem Objektbegriff beschäftigen

"Objektorientierung" ist seit mehr als 20 Jahren eines der meistgebrauchten Schlagworte in der Softwaretechnik. M. Broy und J. Siedersleben haben kürzlich den damit verbundenen Denkansatz einer kritischen Prüfung unterzogen und sind zu einem eher ernüchternden Urteil gekommen [B-S 02]. Dort wird allerdings fast ausschließlich der Bereich der *objektorientierten Programmierung (OOP)* angesprochen, während sich die im Titel angekündigte Ausdehnung auf die "Software-Entwicklung" auf einige eher allgemein gehaltene Aussagen in einem abschließenden Abschnitt beschränkt.

Gerade auf dem Gebiet der *objektorientierten Analyse (OOA)* - die der Programmierung vorausgeht und die u.a. den Bezug zwischen den Aufgabenstellungen aus der "Realwelt" und ihren Software-Lösungen herstellen soll, halten wir eine solche kritische Prüfung für mindestens ebenso angebracht - wenn nicht noch dringlicher:

- Was heißt es, wenn "Software-Objekte Realwelt-Objekten nachgebildet" (*"are modeled after ..."* [C-W 98]) werden sollen, welcher Art sind diese Objekte, wie findet man sie und wie hängen sie zusammen?

- Was bedeutet z.B. eine Anweisung "**`delete Brown`**"? Soll hier schlicht ein Datensatz (etwa für den Kunden *Brown* in einer Kundendatei) gelöscht werden oder geht es um eine Aufforderung zur Liquidation des körperlichen "Objekts" *Brown* oder gar zur gänzlichen (körperlichen und geistigen) Tilgung im Sinne einer Orwell'schen "Vaporisierung"?

- Wie verträgt sich der (anscheinend?) auf Dinge zentrierte, im Grunde statische "objekt-orientierte" Ansatz mit den Anforderungen an eine Software-Entwicklung, die konkrete Probleme der Lebenswelt mit aller ihrer Dynamik und Handlungsvielfalt möglichst "ganzheitlich" lösen soll?

Diese und ähnliche Fragen haben wir in zwei Arbeitskreisen (zur OOA und zur Begriffsbildung in der Softwaretechnik) in den letzten Jahren ausgiebig diskutiert und Zwischenergebnisse daraus regelmäßig veröffentlicht (siehe z.B. [H-M 98], [BHA+ 00], [H-B 01]). Der vorliegende Aufsatz soll diese Diskussion fortsetzen.

"Gegenstand" - ein Universal-Designator

(1) Ein *Gegenstand* ist "etwas uns Entgegenstehendes", d.h. im Prinzip jeder mögliche Fokus unserer Wahrnehmung, Betrachtung oder sonstiger Handlungen einschließlich der Fiktion. Die Frage, ob es "Objekte" jenseits unserer Wahrnehmung[1] gibt, ist müßig - gerade durch unsere erstmalige Wahrnehmung (und sei diese auch nur fiktiv) und Bezeichnung werden sie (automatisch) zum Gegenstand (unserer Betrachtung). In diesem sehr allgemeinen Sinne ist Gegenstand (oder "Objekt im weiten Sinne") zunächst nichts weiter als der Fokus einer "deiktischen" (Sprach- oder sonstigen Zeige-) Handlung, "das, auf das hingewiesen wird" oder kürzer: "das Bezeichnete". B. Russell sagt dazu im Vorwort der engl. Ausgabe des *Tractatus* [Wit 61]:

> "... This amounts to saying that 'object' is a pseudo-concept. To say 'x is an object' is to say nothing. It follows from this that we cannot make such statements as 'there are more than three objects in the world', or 'there are an infinite number of objects in the world'. Objects can only be mentioned in connexion with some definite property. We can say 'there are more than three objects which are human', or 'there are more than three objects which are red', for in these statements the word 'object' can be replaced by a variable in the language of logic, the variable being one which satisfies in the first case the function 'x is human'; in the second the function 'x is red'. But when we attempt to say 'there are more than three objects', this substitution of the variable for the word 'object' becomes impossible, and the proposition is therefore seen to be meaningless. ..."

D.h. hier ist "Gegenstand" weder ein "Konzept" noch ein "Prädikator" - nicht einmal ein "Leer-Prädikator" (Lorenzen), sondern eine Art *Universal-Designator* oder sprachlicher Zeige-Operator, der aus sprachergonomischen Gründen verwendet wird: Statt zu sagen: *"Dies ist rot"* können wir (mit etwas mehr Nachdruck) sagen: "Dieser Gegenstand ist rot".

Objekt = aufgefasster und gedeuteter Gegenstand

(2) 'Gegenstand' ist Teil der deiktischen Sprachgeste, verweist allerdings auf etwas, was jenseits der Sprache liegen kann. Damit dient es (wie andere hinweisende Partikel und Bezeichner) dazu, Nicht-sprachliches sprachlich verfügbar zu machen. So wie das Wort "Gegenstand" ein "General-Bezeichner" für jedwedes, nicht näher spezifiziertes Etwas ist, können wir durch die Wahl spezi-ellerer Bezeichner die Menge der bezeichneten Dinge einschränken - bis hin zur Nullmenge. F. de Saussure hat diese Beziehung wie folgt dargestellt:

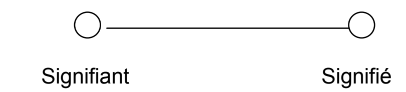


Abb. 1: “Bezeichnungs”-Beziehung nach F. de Saussure

(3) Um zu einem "Konzept" zu werden, muss etwas Wahrgenommenes bezeichnet und aufgefasst *("konzipiert")* werden. Dazu gehört u.a. seine Abgrenzung und "Erschließung", d.h. eine Beschreibung seiner Eigenschaften und Merkmale. Eine solche *Auf*-Fassung (*"conception,"* vgl. FRISCO [FHL+ 98]) führt Unbekanntes auf Bekanntes zurück und ist damit notwendigerweise reduktionistisch: Sie "erschließt" Neues mit dem (begrenzten) Vorrat von Wissen über Bekanntes

[1] *Wahrnehmung* (engl.: *perception*) wird hier als bloße unreflektierte Unterscheidung eines vom anderen und als Vorstufe zur *conception* ("als Etwas erkennen") verstanden (vgl. [FHL+ 98]).

und setzt es dazu in Beziehung [Gha 03]. Sie ist immer endlich und kann die potentiell unendlich vielen Eigenschaften des Gegenstands nur eingeschränkt erfassen. Sie ist auch nicht eindeutig, d.h. zum gleichen Gegenstand können beliebig viele unterschiedliche Konzeptionen existieren.

Wenn wir den Unterschied auch sprachlich deutlich machen wollen, könnten wir die Koexistenz der beiden Worte "Gegenstand" und ""Objekt" in der deutschen Sprache nutzen: *Objekt* ist dann ein *aufgefasster Gegenstand*, d.h. einer, den man nicht nur bezeichnen, sondern auch durch Eigenschaften und Merkmale beschreiben kann.

Triadische Zeichenbegriffe in der Semiotik und Informatik

(4) Hier kann man den von Pierce und anderen Autoren vorgeschlagenen triadischen Zeichenbegriff nutzen (vgl. [FHL+ 98]) : Peirce unterscheidet *sign, object* und *interpretant*: Ein Zeichen (*sign*) kann ein Symbol, eine Zeigehandlung oder ein als solches gedeutetes Phänomen (wie z.B. ein Rauch"zeichen") sein. Es weist auf etwas "Bezeichnetes" (*"object"* im Sinne vom o.g. "Gegenstand") hin. Dazu gehört immer ein *interpretant*, d.h. eine Deutung des Bezeichneten im Kontext des Zeichen-Senders bzw. Zeichen-Empfängers. Der Peirce'sche Interpretant sollte aber nicht mit dem Interpreter (Zeichen-Sender bzw. -Empfänger) verwechselt werden.

Eine Variante der Peirce'schen Triade liefert der FRISCO-Bericht (vgl. Abb. 2). Dort wird anstelle des vielfältig deutbaren *"Object"* das Wort *"referent"* (= das, auf das verwiesen wird) verwendet - mit der an Peirce erinnernden Zusatz-Erklärung *sign object*. Dabei handelt es sich um einen (nicht weiter bestimmten) Bereich (engl.: *domain*). Das von Peirce gebrauchte Wort *sign* könnte insofern missverstanden werden, als man damit womöglich die gesamte Zeichen-Triade ansprechen möchte. Im FRISCO- Bericht hat man deshalb das präzisere *sign token* oder *representation* vorgezogen. Schließlich hat man das Peirce'sche Kunstwort *Interpretant* als *conception* (oder *sign concept* - vgl. (4)) gedeutet.

Konzeptionen und Repräsentationen sind nicht zu trennen von den Individuen bzw. Gemeinschaften, die diese nutzen, sei es, um mit Hilfe von Zeichenprozessen etwas mitzuteilen oder empfangene Mitteilungen zu interpretieren. Im FRISCO-Report hat man dem durch die Erweiterung zum "Semiotischen Tetraeder" Rechnung getragen, bei dem der (menschliche oder - in bestimmten Fällen automatisierte -) *"Aktor"* in der Mitte steht (vgl. Abb. 4).

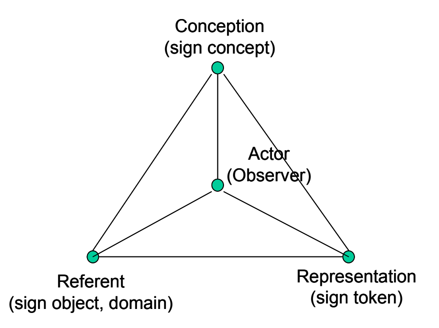


Abb. 2: Semiotischer Tetraeder nach FRISCO

(5) In der Informatik haben wir es mit verschiedenen Arten und Formen von Objekten zu tun. "Objekt" wird hier (fast) immer spezifischer aufgefasst als ein bloßes Synonym zu "Gegenstand" im Sinne von (1) und (2). Elementare Objekte der (herkömmlichen) Programmierung sind die sog. *Konstanten* und *Variablen*: Sie tragen einen *Bezeichner* (*identifier*) bzw. eine *Adresse* und haben einen Bezug, den *Variablen*- oder *Konstantenwert.* Werte sind selbst wieder Bezeichner - entweder für andere Variablen oder Konstanten (das kann man z.B. nutzen, um damit Referenz-Ketten aufzubauen) oder aber sie sind für Programmbenutzer (menschliche *Aktoren*) bestimmte Zeichenreihen (z.B. Texte oder Ziffernfolgen), die diese (bei gelingender Interpretation) womöglich mit Gegenständen ihrer Lebens- oder Gedankenwelt verknüpfen können.

Die Rolle des Interpretanten nimmt in diesem Falle die Interpretationsvorschrift ein, die es z.B. einem Compiler erlaubt, den Variablen- oder Konstantenwert bestimmungsgemäß - etwa nach seiner Zugehörigkeit zu einem bestimmten *Typ* - abzuspeichern bzw. zu interpretieren

Herkömmliche Programmiersprachen haben einen Satz festgelegter Typen - damit ist der Handlungsraum für die "Objekte" (d.h. Variablen und Konstanten) durch einen Satz vorbestimmter, standardisierter Operationen (wie Zuweisungen, Vergleiche, Arithmetik etc.) festgelegt. Bei der *objektorientierten Programmierung (OOP)* wird dieser Handlungsraum (innerhalb bestimmter Grenzen) *programmierbar*: An die Stelle des vordefinierten Typs tritt die vom Programmierer individuell konzipierte *Klasse*, die alle zugehörigen ("gleichartigen") Objekte gemeinsam charakterisiert. Diese enthält nicht nur Aussagen darüber, welche Werte ein Objekt hat oder annehmen kann sondern auch, welche Operationen man darauf anwenden kann, d.h. welche Handlungen man damit vornehmen kann.

Technisch gesehen ist jedes solche "Objekt" ein Speicherbereich - mit einer (Speicher-) Adresse und einer Reihe von Speicherzellen, die die Ausprägungen der o.g. Charakteristika in geeigneter Form als "Wert" repräsentieren. Damit wird die gesamte Triade zum Teil der symbolischen, Computer-verarbeitbaren Welt:

> "An object represents a component of the Smalltalk-80 software system. .. An object consists of some private memory and a set of operations." [G-R 83]

Zeichentriade bei der Objektorientierten Analyse

(6) Eine neue Situation schafft in dieser Hinsicht die seit etwa 1990 propagierte *objektorientierte Analyse (OOA).* Sie erhebt den Anspruch, über den symbolischen Bereich hinauszugehen:

> "You can look around you now and see many examples of real-world objects: your dog, your desk, your television set, your bicycle. These real-world objects ... all have state, and they all have behaviour. ... Software objects are modeled after real-world objects in that they, too, have state and behaviour. ..." ([C-W 98], p. 40)

Damit wird eine Repräsentations-Beziehung (*"are modeled after ..."*) zwischen sog. *real-world objects* und *software objects* unterstellt. Natürlich wurde eine solche Beziehung auch schon vor dem Aufkommen der OOA oft unterstellt, doch sind die damit auftretenden terminologischen Probleme mit der OOA erst richtig deutlich geworden. Können wir mit dem oben ausgeführten triadischen Ansatz auch Realwelt-Objekte im Sinne der OOA angemessen erfassen?

(7) Ein "Bezeichnetes" (sign object) muss nicht notwendigerweise der symbolischen Welt angehören - es kann auch "draußen" - jenseits der Grenzen von Computer und Sprache

angesiedelt sein (vgl. (3)). Nur: was ist genau gemeint, wenn wir eine sprachliche Repräsentation (ein *sign token*) benutzen, um damit einen *Gegenstand der Realwelt* zu bezeichnen? Prinzipiell ist der Gebrauch eines *sign token* mit den gleichen Unsicherheiten behaftet wie das Zeigen mit dem Zeigefinger, mit einem Augenaufschlag oder dem Gebrauch von Wendungen wie *"Dies ist .."* oder *"Dieser Gegenstand ist .."*. Allesamt sind dies Zeigehandlungen, die der Fokussierung auf ein "Gemeintes", aber nicht notwendigerweise dessen eindeutiger Bestimmung und schon gar nicht dessen vollständiger Erfassung dienen können.

Die "vollständige Erfassung" eines Realwelt-Gegenstands (*"real world object"*) kommt allein schon deshalb nicht in Frage, weil diese immer ein endloses Unterfangen wäre: Über jeden Gegenstand - und sei er noch so banal - lassen sich prinzipiell unendlich viele zutreffende Aussagen machen. Hier kommt wieder die 3. (obere) Ecke unserer Triade ins Spiel: Wir können einen solchen Gegenstand zwar nie vollständig erfassen, aber ihn beliebig genau charakterisieren - und zwar durch endlich viele Zuschreibungen (oder "Prädikatoren") - und das ist genau die Rolle einer *Auffassung (conception).* Also: zu einem unendlich charakterisierbaren Gegenstand können beliebig viele endliche Charakterisierungen (*conceptions*) existieren, die jeweils eine *Sicht* auf diesen Gegenstand darstellen. Jede dieser Charakterisierungen kann in einer bestimmten Sprache *repräsentiert* werden. Diese Beziehung muss nicht ein-eindeutig sein, d.h. für die gleiche Charakterisierung können mehrere, verschiedene Repräsentationen benutzt werden und umgekehrt kann eine Repräsentation unterschiedlich gedeutet werden, d.h. für verschiedene Auffassungen stehen.

Die drei Aspekte von "Objekt"

(8) Wenn wir also einen erweiterten, triadischen Objektbegriff im Sinne von (6) bzw. (7) zugrunde legen und dabei jede Art von Bezügen (*referents*) einschließlich solcher auf Realwelt-Objekte zulassen wollen, so können wir dies durch die Unterscheidung dreier *Aspekte* tun, die jedem Objekt zukommen (vgl. Abb. 6): Der ***U-Aspekt*** ist universal, bezieht sich auf einen Gegenstand als Teil eines *Universe of Discourse* und umfasst alles, was man prinzipiell über diesen Gegenstand wissen kann. Damit ist dieser Aspekt potentiell unendlich und durch sprachliche Mittel (einschließlich Computer) womöglich nicht vollständig erfassbar. Mit dem U-Aspekt von Realwelt-Objekten verhält es sich ähnlich wie mit den irrationalen Zahlen (z.B. der Zahl *pi* oder dem nicht-rationalen Grenzwert einer mathematischen Folge oder Funktion): Man kann sie beliebig genau repräsentieren, aber niemals vollständig erfassen.

Will man einen Gegenstand sprachlich "erfassen", so muss man ihn (durch sprachliche Aussagen) charakterisieren ("auffassen" im Sinne von FRISCO, vgl. (4)). Solche Charakterisierungen sind immer endlich, aber in der Regel weder eindeutig noch vollständig. Wir wollen sie als ***C-Aspekt*** eines Objekts bezeichnen.

Für die gefundenen Charakterisierungen lassen sich nun sprachliche Repräsentationen finden - z.B. in Form von Symbolen, Texten, Zahlen, Graphiken, Tabellen - kurzum in jeder Form von Daten. Wir wollen diesen (rein syntaktischen) Aspekt unserer Objekte ***R-Aspekt*** nennen.

Als Beispiel aus einem Informationssystem nehmen wir den Kunden *Brown* (vgl. Abb. 3): Sein U-Aspekt umfasst alles, was wir aus der unendlichen Mannigfaltigkeit seiner physischen und geistigen Existenz über ihn zusammentragen könnten (aber niemals erschöpfend tun können!).

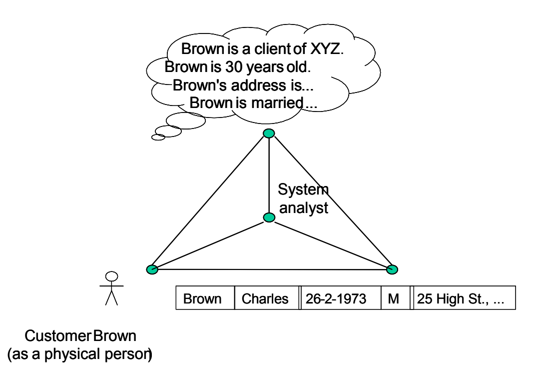


Abb. 3: Beispiel: Die 3 Aspekte eines Objekts in einem Informationssystem

Ein C-Aspekt kann z.B. aus der Menge von Aussagen bestehen, die in Abb. 3 oben stehen. Ein möglicher R-Aspekt dazu könnte eine Zeile in einer Datenbank-Tabelle (aber ebenso eine textuelle Personenbeschreibung, ein Bild oder eine sonst geartete Repräsentation) sein.

Auf Repräsentationen kann man - z.B. als Werte in einem OO-Programmsystem - selbst wieder Bezug nehmen. Damit lassen sich Ketten von Triaden bilden wie in Abb. 4 angedeutet. Hier beziehen wir uns mit dem UML-Konstrukt Brown: Customer auf den Datenbanksatz aus Abb. 3, die Interpretationsvorschrift liefert das entsprechende DB-Schema.

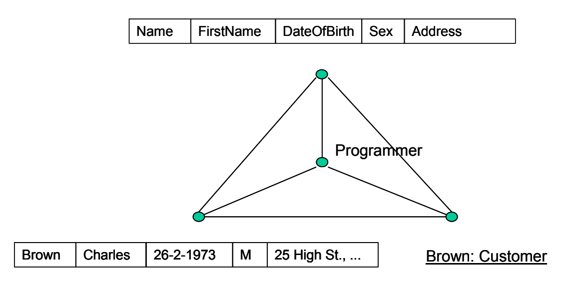


Abb. 4: Beispiel eines Software-Objekts

(9) Zu einer weiteren Deutung des U-Aspekts gelangen wir, wenn wir die Idee des "Einkreisens" eines Gegenstands durch fortlaufende Charakterisierungen aufnehmen und uns einen "Grenzübergang" analog zur Infinitesimalrechnung in der Mathematik vorstellen. Als *Fixpunkt* einer solchen Folge von Charakterisierungen könnte man dann vom *Ding an sich* sprechen. Eine solche Redeweise würde zum einen den in der Informatik (aus guten Gründen) sehr verbreiteten "naiven Realisten" eine Brücke bauen und zum anderen nicht mit der guten konstruktiven Tradition brechen: Denn dieses "Ding an sich" ist eben nicht "da" und muss nur "gepflückt" werden ([Mey 88]), sondern entsteht im Zuge von fortlaufenden (im Grunde nie endenden) konstruktiven Charakterisierungs-Handlungen.

Will man dieses "Ding" (aller prinzipiellen Unmöglichkeit zum Trotz) darstellen, so bedient man sich eines Stellvertreters oder *Surrogats* (wie z.B. des Strichmännchens in Abb. 3 oder der internen Objekt-Id bei objektorientierten Datenbanken). Damit bildet man (gedanklich) eine weitere Zeichentriade, die nun das Surrogat (als Repräsentation) mit dem eigentlichen Ding über die Konzeption des Zeige- und Ersetzungsvorgangs verbindet (vgl. dazu auch [Sow 02]).

Folgerungen für die Software-Entwicklung und ihre "frühen Phasen"

(10) Welche Folgerungen können wir daraus für die Informatik und im besonderen für die sog. "frühen Phasen" der Software-Entwicklung ziehen? Wir fassen zusammen:

- Der Objektbegriff der Informatik (und speziell deren sog. "OO-Techniken") geht über den eines Universal-Designators oder Russell'schen Pseudo-Konzepts hinaus. Objekte sind auch keine natur- oder metaphysisch gegebenen Entitäten, sondern soziale Konstrukte der Beteiligten, die deren Vorstellungen und Intentionen widerspiegeln. Sie sind nicht vorgefertigt, einfach "da" oder "zum Pflücken", sondern müssen von System-Eignern, -Benutzern und -Entwicklern gemeinsam im Diskurs entwickelt - eben *"konstruiert"* werden (vgl. [BHA+00]).

- *OO-Objekte* (wir nutzen diese im Grunde pleonastische Bezeichnung zur Unterscheidung von anderen Objekt-Auffassungen) lassen sich durch die 3 Aspekte *Repräsentation* (Bezeichner bzw. Adresse), *Konzeption* (Interpretation, intensionale Beschreibung) und *Bezug* (Wert, Extension) fassen. Dies gilt in gleicher Weise für Software-Objekte der OOP und für die sog. "Realwelt-Objekte" der OOA:

- Aus dieser Auffassung von Objekten ergeben sich klare Konsequenzen für den Software-Entwicklungsprozess: Am Anfang der Software-Entwicklung stehen keine fertigen Objekte, sondern Wahrnehmungen, Konzeptionen und Beschreibungen von Sachverhalten und Handlungen, die es zu verstehen, zu bearbeiten und im gemeinsamen Diskurs zur Deckung zu bringen gilt. Objekte können daraus als Brennpunkte des gemeinsamen Verständnisses herausgearbeitet werden - im ständigen Wechselspiel zwischen den Aspekten Bezugsbildung - Konzeption und Interpretation - Repräsentation. Moderne OO-Techniken unterstützen - zumindest teilweise - diesen Prozess: So geht es bei der für die OOA empfohlenen Anwendungsfall-Analyse (*use case analysis*, vgl. [Jac 93]) zunächst darum, Sachverhalte, Vorgänge, Handlungen im Bezugsbereich zu untersuchen, diesen dabei näher abzugrenzen und dann erst zu Konzeptionen, Repräsentationen und schließlich zu Objekten zu kommen. Ebenso beruht der Erfolg der Datenabstraktionstechnik (einer wesentlichen Grundlage für die OOP) nicht zuletzt auf der Einsicht, dass zentrale Konzepte wie Typen und Klassen primär nicht aufgrund struktureller Eigenschaften, sondern als Handlungspotentiale (= Mengen anwendbarer Operationen) zu definieren sind.

- Die vorherrschende Ansicht, man könne die (OO-) Objekte aus den Anwendungsfällen einfach durch Anstreichen der Substantive und Sammeln in einer "Objektliste" (vgl. [Jac 93]) herausholen, greift dagegen zu kurz. Hier können z.B. Begriffs-basierte Verfahren wesentlich mehr Hilfestellung leisten (vgl. [D-H 00]).

- Objektorientierte Analyse und Modellierung ist somit keine "Abbildung" unverrückbarer, von vornherein feststehender "natürlicher" Sachverhalte, sondern interessen-geleitete Konstruktionsarbeit von Akteuren - eben den Protagonisten der Software-Entwicklung. Das bedeutet für den einzelnen Akteur, dass sein Modellierungs-Spielraum möglicherweise größer als von ihm selbst angenommen ist. Was diesen begrenzt, sind oft weniger die (vorgeschobenen) "Sachzwänge", sondern die Interessen anderer (sichtbarer oder verborgener) Akteure.

Dank: Allen Mitgliedern und Mit-Diskutanten des Arbeitskreises "Terminologie der Softwaretechnik" sowie des Münchner Arbeitskreises zur OO-Analyse und Modellierung danken wir an dieser Stelle für die langjährige Zusammenarbeit und die zahllosen daraus erwachsenen Anregungen und Ideen.

Literaturhinweise:

[BHA+00] H. v. Braun, W. Hesse, U. Andelfinger, H.B. Kittlaus, G. Scheschonk.: Conceptions are social constructs - Towards a solid foundation of the FRISCO approach. In: [FLV 00]

[B-S 02] M. Broy, J. Siedersleben: Objektorientierte Programmierung und Software-Entwicklung - Eine kritische Einschätzung. Informatik-Spektrum 25.1, pp. 3-11 (2002)

[C-W 98] M Campione, K. Walrath: The Java Tutorial – Object-Oriented Programming for the Internet. 2nd ed., The Java Series, Addison Wesley 1998

[D-H 00] S. Düwel, W. Hesse: Bridging the gap between Use Case Analysis and Class Structure Design by Formal Concept Analysis. In: *J. Ebert, U. Frank (Hrsg.): Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik. Proc. "Modellierung 2000"*, pp. 27-40, Fölbach-Verlag, Koblenz 2000

[FHL+98] E. Falkenberg, W. Hesse, P. Lindgreen, B.E. Nilsson, J.L.H. Oei, C. Rolland, R.K. Stamper, F.J.M. Van Assche, A.A. Verrijn-Stuart, K. Voss: FRISCO - A Framework of Information System Concepts - The FRISCO Report. IFIP WG 8.1 Task Group FRISCO. Web version: http://www.wi.leidenuniv.nl/~verrynst/frisco.html (1998)

[FLV 00] E.D. Falkenberg, K. Lyttinen, A.A. Verrijn-Stuart (Eds.): Information System Concepts - An Integrated Discipline Emerging. Proc. IFIP TC8/WG8.1 Conference ISCO-4. Kluwer Academic Publishers 2000

[Gha 03] M. Ghasempour: Weisheit als Kritik der totalitären Wissensauffassung. Persönliche Kommunikation, Jan. 2003

[G-R 83] A. Goldberg, D. Robson: Smalltalk 80: The language and its implementation; Addison-Wesley 1983

[H-B 01] W. Hesse, H. v. Braun: Wo kommen die Objekte her? Ontologisch-erkenntnistheoretische Zugänge zum Objektbegriff. In: K. Bauknecht et al. (eds.): Informatik 2001 - Tagungsband der GI/OCG-Jahrestagung, Bd. II, S. 776-781. books_372ocg.at; Bd. 157, Österr. Computer-Gesellschaft 2001

[H-M 98] W. Hesse, H.C. Mayr: Highlights of the SAMMOA framework for object oriented application modelling. Proc. DEXA 98 – 9th Int. Conf. and Workshop on Database and Expert Systems Applications

[Jac 93] I. Jacobson et al.: Object-Oriented Software Engineering - A Use Case Driven Approach; Revised Printing, Addison-Wesley 1993

[Mey 88] B. Meyer: Object-Oriented Software Construction; Prentice Hall 1988

[Sow 00] J.F. Sowa: Knowledge Representation: Logical, philosophical and computational foundations. Brooks Cole Publ. Co. 2000

[Wit 61] Wittgenstein, Ludwig: Tractatus logico-philosphicus. Translated by D.F.Pears & B.F.McGuinness, with the Introduction by Bertrand Russell, Humanities Press International, Inc., Atlantic Highlands,NJ, 1961

Objekt und Klasse, Gegenstand und Begriff

Herbert Hrachovec

Institut für Philosophie, Universität Wien

Objektorientierte Vorgangsweisen und philosophische Theorien berufen sich auf die intuitive Weltsicht. Was sie daraus entwickeln, und wie sich die Resultate zueinander verhalten, ist nicht mehr so intuitiv. Im Gegenteil, es ist ausgesprochen schwierig, die beiden Ansatzpunkte richtig ins Verhältnis zu setzen. Sie sind hybrid miteinander verschränkt.

Computerprogramme sind in einer Hinsicht Maschinenabläufe, wie der Benzinantrieb oder Fotokopieren. Die objekt-orientierte Software-Entwicklung setzt einen anderen Akzent. Sie modelliert Probleme des konkreten Arbeitsalltags und findet dazu digitale Lösungen. Wie schon der Name sagt, stützt sie sich auf eine tief verankerte Intuition: die Welt besteht nach allgemeiner Meinung aus Gegenständen im Rahmen von Tatsachen. Und durch vergleichende Zusammenfassung von Dingen ergeben sich Allgemeinbegriffe, mit deren Hilfe Ordnung in der Welt geschaffen wird. Das ist die 2. Intuition im Hintergrund der objekt-orientierten Programmierung. Sie modelliert Strukturkonstanten (und auch dynamische Abläufe) der gegenständlichen Welt als „Klassen“.

Philosophen (m/w) leisten zur Erstellung von Software in der Regel keinen Beitrag, aber dieser Fall liegt aus mindestens zwei Gründen komplizierter. Erstens fallen Überlegungen, woraus die Welt besteht, auch und gerade wenn sie selbstverständlich klingen, in den Fragebereich der klassischen Metaphysik. Sie diskutiert unsere leitenden Intuitionen im Umgang mit der Wirklichkeit. Zweitens ist das Inventar der Objektorientierung jenem der Mengentheorie verwandt, die ihrerseits dazu verwendet wird, logische Termini wie „Begriff“, „Urteil“ und „Schlußfolgerung“ sprachanalytisch zu klären. Die Allgemeinbegriffe der Erkenntnis- und Wissenschaftstheorie sind durch mengentheoretische Konstrukte gut zu erläutern; so legt sich der Gedanken nahe, daß die Beziehung auch für die Softwaretechnik gilt. Ich werde zeigen, daß es sich um zwei voneinander deutlich getrennte Verwendungsweisen von „Klasse“ handelt, die sich allerdings auf interessante Weise ergänzen. Daraus ergibt sich ein Kommentar zur impliziten Metaphysik objektorientierter Vorgangsweisen.

Schablone und Urteil

Im Urteil wird, nach traditioneller Lehre, ein Gegenstand unter einen Begriff subsumiert. „Das ist ein Peugeot 307“ heißt: dieses Ding gehört zu einer bestimmten Menge von der genannten Firma produzierter PKWs. Das Beispiel könnte –mutatis mutandis – aus einer Einführung in objektorientiertes Programmieren stammen. In dieser Welt „spezifiziert eine Klasse die Gemeinsamkeit einer Menge von Objekten mit denselben Eigenschaften (Attributen), demselben Verhalten (Operationen) und denselben Beziehungen.“ [Balzert 1999, S. 113] Also etwa einen Fahrzeugtyp, dem entsprechend einzelne PKWs gebaut werden. Derart charakterisierte Objekte unterscheiden sich u.a. in der Motorstärke, der Farbe und dem benötigten Treibstoff. Jedes verfügt über eine eindeutige Identifikation und vordefinierte Verhaltensweisen. Mit anderen Worten: ein Kraftfahrzeug, das solchen Bedingungen genügt, ist ein Objekt der Klasse „Peugeot 307“. Der Unterschied zum Satz „Das ist ein Peugeot 307“ erscheint minimal. Aber der Schein trügt.

In beiden Fällen wird ein Einzelding in ein Verhältnis zu etwas Allgemeinem gesetzt. Doch die Auffassungen darüber, worin die Beziehung besteht, differieren beträchtlich und wirken sich direkt auf das Verständnis der Verhältnis*glieder* aus. Ein Gegenstand, der einem Urteil unterzogen wird, ist nicht einfach die Realisierung eines Typs. Für die objekt-orientierte Analyse kann das besagte Fahrzeug *als modelliertes Objekt* einzig und allein der angegebenen Klasse angehören. Alle relevanten Charakteristika liegen auf ihrer Seite. Urteile finden hier keinen Anhaltspunkt. In ihnen geht es um die Abschätzung, inwieweit sich bestimmte Charakteristika von Gegenständen behaupten lassen. Dazu muß es experimentelle Handlungsfreiheit geben, ein Ausprobieren von Beschreibungen und Rückkoppelungen zwischen Subjekt und Prädikat. Die Ausprägung eines Attributes im Objekt kann hingegen keine Rückwirkung auf die Definition der Klasse haben. Ein Urteil dient z.B. dazu, nach einer Testfahrt die Verkehrssicherheit des Fahrzeugs zu bewerten. Seine Erfassung als software-technisches Objekt entspricht dem Eintrag im Typenschein. Als Exemplar eines Typus ist es erschöpfend bestimmt.

Auf diesen Hinweis könnte jemand erwidern: „Auch die Typenzuordnung ist veränderbar!“ Sicherlich, aber der Einwand illustriert genau die Divergenz zwischen den beiden Verhältnismustern. Für bestimmte Zwecke soll der Einzelfall durch das vorgegebene Schema definitiv festgelegt sein; in anderen Kontexten kommt es gerade auf den Spielraum zwischen Einzelnem und Allgemeinem an, auf die begriffliche Flexibilität in der Möglichkeit der Musteradaption.

Der Punkt ist nicht, daß ein nach Schablone gefertigtes Produkt keine Geschichte haben könnte, sondern daß es in einem solchen Fall keine einfache Instanz der Schablone mehr ist. Sobald sich die Frage stellen *kann*, zu welchem Typ das Ding gehört, befinden wir uns in einer anderen Diskussion.

Die Attribute des PKW, um beim Beispiel zu bleiben, durchlaufen dann nicht bloß den parametrisierten Raum, sondern leisten einen Beitrag zur *Bestimmung* der Allgemeinheit, der sie subsumiert werden. (Schablonenhaft festgelegte Entwicklungsverläufe treffen auf dieselbe Schwierigkeit.) Wenn man es im Vokabular der „Unified Modeling

Language“ formulieren will: Begriffe sind eher *Anwendungsfälle*, in denen eine Objekt-Klassen-Konstruktion von Personen zur Bewältigung kognitiver Aufgaben herangezogen wird

Begriffsgebrauch bedeutet auch Modifikationen in der Abstimmung zwischen singulären Gegebenheiten und den in Prädikaten artikulierten allgemeinen Mustern. Personen fällen nach Kriterien Entscheidungen darüber, ob Dingen bestimmte Eigenschaften zugesprochen werden. In dieser Praxis ergeben sich dadurch die Spielräume, von denen vorhin die Rede war. Traditionell bezeichnet man die Fähigkeit zu einer solchen Feinabstimmung zwischen Einzelnem und Allgemeinem als Urteilskraft. Mit ihrer Hilfe wird die Vertretbarkeit der Anwendung des Begriffes auf ein Ding im Einzelfall entschieden. „Das ist ein Peugeot 307“ enthält danach eine Behauptung, die eventuell umstritten ist, z.B. wenn es um eine speziell getunte Maschine geht. Dieser gesamte Verhaltenskomplex fehlt in der Objekt-Klassen-Konstruktion. Ein Software-Objekt gehört zu einer Klasse, *doch die wird nicht von ihm prädiziert*. Das Repertoire der Einsatzmöglichkeiten von solchen Objekten unterscheidet sich in wesentlichen Punkten von jenem der (Namen für) Gegenstände, mit denen sprachlich operiert wird. Die intuitive Weltsicht („Dinge, zu Tatsachen verbunden“) wird uneinheitlich ausgelegt. Einmal handelt es sich um Ausprägungen einer Muster-Vorgabe, das andere Mal um Träger von (eventuell wechselnden) Eigenschaften. Die Vorgabe inkludiert eine „object factory“, welche den Eigenschaften fehlt. Im Detail fallen die beiden Verhältnisbestimmungen von Einzelnem und Allgemeinem deutlich auseinander.

Serienproduktion und Welterzeugung

Umgangssprachlich sind die beiden Strategien etwa im Unterschied zwischen Blumentöpfen oder Telegrafenstangen auf der einen, Giftpilzen oder Kopfbedeckungen auf der anderen Seite greifbar. Die automatische Fertigung gehorcht anderen Gesetzen, als die exemplarische Verwirklichung einer Gattung. Das heißt nicht, daß es keine Berührungspunkte gäbe. Die übergeordneten Kategorien – Einzelnes/Allgemeines – in denen das Problem sich fassen läßt, sind offensichtlich hilfreich. Man *muß* den Akzent nicht auf die Unvereinbarkeiten legen. Wir haben Objektstatus gegen Beurteilung abgehoben. Von einem allgemeineren Standpunkt aus handelt es sich in beiden Fällen um *Prüfverfahren*. Das Exemplar einer Klasse gehorcht implizit einem Zugehörigkeitstest, andererseits werden Gegenstände nach Kriterien zu Mengen zusammengefaßt. Beide Vorgänge lassen sich, wie Freges mathematische Rekonstruktion des Begriffsgebrauches überzeugend demonstriert hat, im selben Formalismus fassen.

Begriffe sind danach Funktionen, die es erlauben, Gegenstände zusammenzufassen, das ergibt Extensionen des betreffenden Begriffsausdrucks. Diese Konzeption paßt auf Blumentöpfe ebenso, wie auf Kopfbedeckungen. (Die Intension, welche in den Bereich der inhaltlichen Handhabung von Begriffsausdrücken fällt, kann man für viele Zwecke einklammern.) Damit ist der Punkt erreicht, von dem aus sich die in Aussicht gestellte wechselweise Ergänzung von Objektorientierung und Prädikatenlogik beleuchten läßt.

An eine Softwarekonstruktion und die Überprüfung einer Mengenzugehörigkeit werden unterschiedliche Anforderungen gestellt. Dennoch gilt auch, daß beide Faktoren häufig verschränkt auftreten. Ein Test, der den Objekttyp feststellt, ist in solchen Fällen mit klassischen Urteilsvorgängen kombiniert. Der Umgang mit Strichkodes illustriert die Zusammenhänge plastisch.

An der Kassa eines Supermarkts koexistieren zwei Verfahren von Objektmanipulation. Das eine erfaßt den Gegenstand automatisch als Instanz einer bestimmten Klasse und setzt einen Verrechnungsprozeß in Gang. Das andere ist kundenorientiert und seine Beschaffenheit ist nicht so eindeutig vorgegeben. Es besteht z.B. in der Kontrolle der ausgewiesenen Produktbezeichnung oder der Übereinstimmung von Ware und Verpackung. Die beiden Vorgänge sind nur mit großer Mühe (wenn überhaupt) austauschbar. Ohne die fraglose Zuordnung eines Objekts zu seiner Klasse im Verrechnungsvorgang steht der Betrieb. Aber das Geschäft stockt ebenfalls, wenn Regeln dafür fehlen, wie Reklamationen gegen Fehlverrechnungen zu handhaben sind. Der Kaufvorgang verbindet Automatismus, Urteilsfähigkeit und deren wechselseitigen Abgleich. In Grundsatzdiskussionen werden Typen und Begriffe, Extensionen und Intensionen, Mechanik und Überlegung, gerne gegeneinander ausgespielt. Das Bild, das sich aus der gegenwärtigen Diskussion ergibt, ist differenzierter. Objekte und Gegenstände, bzw. Klassen und Begriffe, stammen aus unterschiedlichen Problembereichen. Eine Abstraktionsstufe höher ist die Differenz auflösbar. Die meisten gesellschaftlichen Prozesse enthalten jedoch eine Mischung aus Steuerung und Argumentation.

Platon für Ingenieure

Dirk Siefkes hat in diesem Zusammenhang von „Hybridobjekten“ gesprochen. [Siefkes 02, passim] Software-Produkte zeigen ein Janusgesicht: auf der einen Seite gesteuerte Prozesse, auf der anderen Handlungen. Ein drastisches Bild wären siamesische Zwillinge. Zwei unabhängige Individuen hängen an einer lebenswichtigen Stelle zusammen. Sie ist für beide identisch, auch wenn sie unter erweitertem Aspekt irreduzibel verdoppelt erscheint. Weniger spektakulär läßt sich die Sache am Beispiel von Straßenbezeichnungen diskutieren. Es ist nicht ungewöhnlich, daß das Teilstück einer Route mehrfache Namen trägt, weil unterschiedliche Linienführungen für diese Strecke zusammenfallen. (Der „Romantikpfad“ ist dann gleich dem „Industriezubringer“.) Die Logik solcher Überlegungen wäre getrennt zu untersuchen.

Damit ist das Verhältnis zwischen Softwaretechnik und Philosophie auf eine Formel gebracht, welche die Spezifika beider Seiten einschließt. Wie Bauelemente eine statische und ästhetische Funktion besitzen können, verbindet eine Kassa im Supermarkt Mustererkennung und Urteilsfähigkeit. So friedfertig soll der Beitrag nicht schließen. Mehrfach ist darauf verwiesen worden, daß sich der Gegensatz zwischen Steuerung und Argumentation unterlaufen läßt. Eine humanistische Variante dieser Strategie ist in der Philosophie populär. Eingangs wurde festgehalten, daß das Paradigma objektorientierter Programmierung die Selbstverständlichkeiten des Alltags zum Ausgang für

die Modellierung nimmt. Wir teilen die Welt in Dinge, Verhältnisse und ihre einigermaßen verläßlichen Eigenschaften. Während die Softwareentwicklung daran anknüpft, finden viele Philosophen (m/w) diese Zustandsbeschreibung ausgesprochen fragwürdig. (Frieder Nake hat am Symposium auf J. Derrida hingewiesen.) Serienproduktion ist für sie die technische Umsetzung des Unterschieds zwischen Typ und Exemplar; der Gebrauch von Begriffen sollte dieser Mechanik nicht assimiliert werden. Die philosophisch populäre These lautet, *Begriffsgebrauch*, und nicht die Umsetzung von Schablonen, sei, trotz der diskutierten Wechselabhängigkeit, der eigentliche Kern des Themas.

Das Argument verläuft in etwa so. Die Welt organisiert sich nicht von selbst in Pilzsorten, Automarken oder Eigenheime. Woraus besteht sie überhaupt? Zu dieser Frage kommen Menschen, weil sie auf Dinge zeigen und ihre Eigenschaften untersuchen können. Die Diskussion, ob und wie ein Gegenstand unverkennbare, unentbehrliche Qualitäten besitzt, durchzieht die europäische Philosophiegeschichte. Eine seit Platon verfügbare Antwort ist in abgewandelten Formen bis heute wirksam geblieben. Um nicht in Einzelbeobachtungen stecken zu bleiben, ist ein Leitbild unerläßlich. Sokrates spricht von der Idee des Schönen:

> Wenn nämlich irgend etwas anderes schön ist außer jenem An-sich-Schönen, so ist es meiner Ansicht nach aus keinem anderen Grund schön, als weil es an jenem Schönen teilhat. ... wenn mir jemand sagt, daß irgend etwas schön ist, entweder weil es eine blühende Farbe oder Gestalt oder sonst etwas der Art hat, so lasse ich das andere auf sich beruhen, denn durch alles übrige werde ich nur verwirrt, und halte ganz einfach und schlicht und vielleicht einfältig daran bei mir fest, daß nichts anderes es schön macht als eben die Anwesenheit oder die Gemeinschaft jenes Schönen, wie und woher sie auch komme. Darüber nämlich möchte ich nichts weiter behaupten, als daß durch das Schöne alle schönen Dinge schön werden. [Platon, Phaidon 100 c,d]

Das Motiv läßt sich auch auf Automarken anwenden. Eine moderne Paraphrase Platons könnte folgende Überlegungen enthalten:

> Wenn mir jemand sagt, daß etwas ein Peugeot 307 ist, entweder weil es eine bestimmte Farbe oder Gestalt, oder sonst etwas der Art hat, so halte ich ganz einfach daran fest, daß nur die Typenzugehörigkeit etwas zu einem Peugeot 307 macht.

Wir sehen etwas unter dem Einfluß vorweg bestehender Kenntnisse. Begriffe artikulieren Typologien, ohne welche uns die sensorischen Inputs überschwemmen. In dieser Theorie stehen hinter den Urteilsakten Urtypen. Die Frage, woraus die Welt besteht, ist äquivalent zur Frage, wie sich mit Hilfe vor-investierten Wissens die Wahrnehmungsdaten zu relativ stabilen Aggregaten ordnen lassen. Ironischerweise bedient sich die objektorientierte Analyse also eines antiken Schemas, dessen Plausibilität zur Modellierung des Erkenntnisvorgangs vielfach angezweifelt worden ist. Aktuelle Konzeptionen vertreten die Auffassung, daß sich die Orientierung in der Welt, genau gesagt die uns zugängliche Welt selber, zusammen mit dem Begriffsgebrauch entwickelt. Das heißt: in der sprachlich vermittelten Strukturierung der Umwelt an Hand von Aussagesätzen. Die Prioritäten Platons werden dabei umgedreht: am Anfang steht kein überzeitlicher Typus, sondern der im diskursiven Prozeß verankerte Entwurf von Ordnungsstrukturen.

Die Klassen, die in der objekt-orientierten Analyse konstruiert und in Programmabläufe eingebaut werden, beruhen auf Sichtweisen einer gegliedert erschlossenen Umwelt, über die wir Behauptungen aufstellen. Kein Strichkode ohne Käuferinnen. Objektorientierte Softwareentwicklung modelliert ein Ensembel platonischer Ideen und gewinnt daraus effektive Abläufe. Ohne derartige Mechanismen ist, wie gesagt, Leben schwer vorstellbar. Doch wenn es dabei bleibt, lohnt es sich auch nicht. Der Zweck der Übung sind Erfahrungen, deren Leitlinien ihrerseits in Frage gestellt werden können.

Literatur

[Balzert 99] Helmut Balzert: Lehrbuch Grundlagen der Informatik. Konzepte und Notationen in UML, Java und C++. Algorithmik und Software-Technik. Anwendungen. Berlin - Heidelberg. 1999

[Booch 99] Grady Booch, Jim Rumbaugh, and Ivar Jacobson: Das UML-Benutzerhandbuch 1999. Bonn

[Frege 75] Gottlob Frege: Funktion, Begriff, Bedeutung, 1975. Göttingen

[Platon 74] Phaidon, Wissenschaftliche Buchgesellschaft, WW 3, 1974. Darmstadt

[Siefkes 02] Dirk Siefkes: Hybridization in Computer Science. TU Berlin, Fak. Elektrotechnik & Informatik. Bericht 02-17

A Remark on Stamper's Dissenting Position on the FRISCO Report

Roland Kaschek

Massey University, Palmerston North, New Zealand
Roland.Kaschek@ieee.org

Abstract. The FRISCO process resulted in a mentalistic foundation for information systems. However, not all of the co-authors of the FRISCO report shared the respective opinion. *Ron Stamper* in a dissenting position rejected several of the fundamental FRISCO postulates and replaced them by others. In particular he postulated information systems being founded on semiosis, i.e., processes of sign usage and rejected mentalistic concepts used in the FRISCO report. The present paper argues that a mentalistic approach is compatible with founding information systems on semiosis. It argues further that surrender of mentalistic concepts leads to losses concerning topics that by information systems research must be understood in total. Examples of such topics are the systems development process and the concept of model.

1 Introduction

Scanning recent texts on information systems reveals that the discussion of their fundamentals until now has not come to an end. "During the last three decades the concept of information system and the discipline of information systems underwent an evolution ..." as is said in [6, p. 1]. To give evidence for this statement the source lists several definitions of the term information system. These range from the 1974 definition of *Mader* and *Hagin* as systems providing "... transaction processing and decision support ..." (cited after [6, p. 1]), over the 1982 definition from *Brookes* et al. as "... all forms of information collection, storage, processing and communication ..." (cited after [6, p. 1]), and the 1995 definition from *Tatnal* et al. as "... [a system] comprising hardware, software, people, procedures, and data, integrated with the objective of collecting, storing, processing, transmitting and displaying information" (cited after [6, p. 1]) to the 1988 definition of *Sandstrom* as "... a designed tool, the purpose of which is to serve people in active work with information and in an organization. It is an organized construction with subsystems for collecting, processing, storing, retrieving and distributing information ...".

The present paper approaches a problem left open by the FRISCO process the aim of which was developing a consistent position concerning the fundamentals of information systems. For documentation of the process see, e.g. [14, 15]. The problem is whether it is reasonable to employ concepts addressing mental aspects for the foundations of information systems. The respective discussion at least goes back to the FRISCO process and I take as a starting point *Ron Stamper's* dissenting position in [15, 191-6]. *Stamper* proposes alternatives to four of the six general assumptions made in [15, p. 29 - 30]. I do not discuss all of them but only the last one saying that information systems can be studied without employing mentalistic concepts such as perception, conception, etc.

From a holistic perspective on information systems *Stamper's* position might appear not feasible since discussion of information systems to a large extent must focus on the social

sphere in which these systems will be fitted after their development. One of the key points here is whether system developers can make customers use the new or adapted system and pay for the respective effort. This is the problem, not to build just a system but a proper or suitable one. Whether a system is proper or suitable in part is up to negotiation. However, in part it is a system quality issue. *Stamper's* dissenting position thus seems to imply to not deal with quality issues or deal with them in a way that avoids reference to mentalistic concepts. I believe that the latter is impossible: Developers need to forecast the customer behavior, e.g. regarding satisfaction with the deployed system and willingness to use and pay for it. The language most commonly and efficiently used for this is the language of everyday psychology which we use to ascribe mental states to each other and based on these states forecast behavior. Using this language by no means implies an objectivistic position concerning the mind. It simply implies the use of a successful causal model of how humans behave. Causal behavior models help us in case of failure finding a new approach to success.

The point that needs to be discussed to clear up things related to *Stamper's* position is quality. My thesis in this respect is:

(i) quality issues cannot fully be discussed without involving mentalistic concepts, and
(ii) inability to understand system quality implies information systems research and practice capitulating in front of solvable problems.

Outline In section 2 I discuss *Stamper's* position. In section 3 the linguistic foundations of the present paper are introduced. Quality aspects are discussed in regard of information systems development in section 4. Quality aspects in regard of models concerning information systems are discussed in 5. The paper is concluded with the resume in section 6.

2 Stamper's dissenting position

Linguistic foundations of information systems were discussed by the FRISCO project, see [15]. However, it did not address consequences for system development. Neither in the glossary nor in the index can an entry for quality be found. The dissent of the majority of the FRISCO team with *Stamper* -as I believe- could not be resolved since the yardstick for this resolution: quality was not consensual and not even reasonably well discussed. Arbitrary definitions of quality could be used. I prefer a definition of it that is in line with the ISO/IEC definition, see [18]. I define with regard to an entity T the quality of T as the consistency of T that makes T suitable for its intended (or actual) use. The definition of quality in [30] shows that the present definition of quality fits philosophical definitions. However, and this is its advantage compared to the more general philosophical definition, it is more specific in that it relates quality to usage. The advantage of doing so is that quality is put into a context of actual or immediate life and subject to a human yardstick. Relating quality to something different easily may lead to either conceptual difficulties (e.g. if quality would be related to a Lord's perspective) or being questioned as inappropriate (e.g. if quality would be related to legal prescriptions, traditions, etc. only). For a discussion of the related concepts 'qualia' and 'qualities' see, e.g. [16]. What I address here with the concept of quality significantly overlaps with that what is addressed by *Boehm* ([9]) and *Boehm, Guo Huang* ([10]) as value based software engineering.

Stamper, see [15, p. 192], says in his dissenting position on the FRISCO report: "We can study information systems without using such mentalistic notions as perceptions, conceptions, senses, mental states, cognitive and intellectual processes; we need only study overt regularities of behavior, but with special emphasis upon the uses made of some things to signify others." Mentalism, according to [30] is an approach to philosophy also present in psychology and linguistics that does not, as required by Behaviorism, restrict itself to externally visible behavior. Behaviorism again according to the source does not count the inner world of an individual as belonging to the things science deals with. I call concepts mentalistic if they conceptualize this inner world. I believe we need mentalistic concepts when we want to forecast human behavior effectively. With behaviorism we can come to presuppose an inner parameter governing the externally visible behavior of humans. Decomposing this inner parameter, classifying its instances and conjecturing its inner relations may be outside the scope of behaviorism. I believe it helps understanding and forecasting human behavior. I don't intend to "prove" that *Stamper* is wrong since most likely one wouldn't be able to agree on a truth theory to be presupposed, a formalization of the question under discussion and a logic to rule out invalid inferences.

However, it is easy to address certain aspects in information systems research based on mentalistic concepts while it is difficult to deal with these aspects without using mentalistic concepts. I discuss two such aspects of information systems research: Quality and models. I discuss quality since information systems are supposed to meet a purpose and thus need to be ascribed a particular quality. Only if customers do so will they be willing to pay for the development effort. I discuss models since models in the area of information systems are widely used and appear to be indispensable. I use *Lockemann and Mayr's* ([24]) theory about what a model is, namely an imagination and apply to it *Stachowiak's* general model theory. Models (in the sense of the present paper) are considered as useful for a particular individual, a particular task and analysis, and for particular period in time only. *Stachowiak*, see [34, 33, 32], considered this quality to be present if a model has the mapping property (Abbildungsmerkmal) implying that a model is always a model of something, its original; truncation property (Verkürzungsmerkmal) implying that the modeler in general will not ascribe all the properties ascribed to the original to the model as well; and the pragmatic property (pragmatisches Merkmal) implying that using the model can only be justified with respect to particular individuals using it, a particular task and analysis, and for a particular period in time.

I use a linguistic approach as a foundation for information systems not only because I believe it is preferable to do so but also because I want to provide a base of discussion that *Stamper* would accept. I'm not going to criticize the sign concept (a definition is given below) or *Stamper's* understanding of it. However, I believe that —much in the line with speech-act theory (, see for it, e.g. [4])— a thorough study of sign usage would need to focus on successful or proper sign use. Study of sign use would involve judgments about at least the following aspects of reference:

- was a proper thing referred to, and
- was the thing referred to referred to in a proper way.

I believe that discussion of these items needs to involve consideration of a referring culture that defines the quality of acts of referring as well as the quality to be achieved in such acts. I believe this cannot successfully be done without employing mentalistic concepts. A definition

of the sign concept suiting me thus would need to take into account acts of referring and the culture governing these acts including judgments about such acts.

Though I generally agree with most of what *Stamper* says I mention that I do not agree with his opinion that we do not have empirical access to conceptions ([15, p. 194]). A source that in some detail shows that mental constructs can be empirically investigated is [12]. I further do not agree with his opinion on how signs are introduced. A position very similar to his is developed in [20]. I do not agree with *Stamper* ([15, p. 193]) who believes that he -by giving examples of signs- can make an individual reading his dissenting position understand the sign concept. Instead of this I believe that for a finite set S of things concepts C, C' can always be constructed such that $C \neq C'$ and that $S \subseteq Inst(C), Inst(C')$ holds where $Inst(C), Inst(C')$ respectively are the instance set of C and C', i.e., the sets of things that respectively fall under C and C'. Consequently I believe the sign concept cannot be introduced by mere examples. The argument represented in more detail is like this: Let $T = \{T_1, \ldots, T_n\}$ be a finite set of things and C be a concept. If $Inst(C)$ does not contain all things then there is a thing X not falling under C and one can define the concept C' as "the concept of all things that fall under C or that are equal to X". Obviously the required relations hold. If on the other hand $Inst(C)$ contains all things then one can identify $X \in Inst(C) \setminus T$ and flip the roles of C and C'. This argument implies that referring to something is a quite sophisticated thing to do and that a respective culture needs to be acquired in order to ensure a satisfactory likelihood of referring properly.

3 Linguistic Foundations of IS

An early definition of the concept information system according to [17, p. 11] is due to the Scandinavian school of information systems. This school headed by *Langefors* defined information system "... [f]rom a functional perspective (...)" as "... technologically implemented media for the purpose of recording, storing, and disseminating linguistic expressions as well as for the supporting of inference making ...", see [17, p. 11]. Another, complementary perspective on information systems that employed a structural view on these systems according to [17, p. 11] was already used by *Davis* and *Olsen* in 1985 and perceived information systems as "... collection of people, processes, data, models technology and partly formalized language, forming a cohesive structure which serves some organizational purpose or function."

According to [28, p. 628-9] a sign is a pair of a signifier and a thing signified. The signifier, also called sign token, stands for the signified thing. I consider the linguistic expressions addressed in the functional perspective definition of information systems as composite signs. A consequence of this refined functional perspective definition of information systems is that customers and the technical component of an information system, i.e., its artifact exchange composite signs also called messages. Following the definition of communication by *Liska* and *Cronkhite* as "the exchange of certain types of signs", compare [11, p. 5] this exchange is communication. Understanding the usage of information systems as communication in this way is what I denote "linguistic usage model".

The signifiers of the signs that are exchanged between information system and customer and the thing they signify were defined during system development. Due to the respective education and usage experience the customer is aware in principle of the signified thing, i.e., the so-called meaning but in a specific stage of system use might mix up things signified

by different sign tokens or might refer in an improper way to the signified thing. At least two different ways of reference to a signified thing are in use in information systems research and practice: a descriptive (used to describe a domain by a model of it) and a prescriptive one (used to prescribe an implementation by a design). For more information about ways of reference see [22].

Referring back to work in [25] I use the metaphor of information space that an information system creates as a conceptual framework for understanding what customers do while using information systems. This space consists of locations at which information objects are located and connections between these. Customers can position themselves on a location in information space, locate information objects, navigate through this space, and interact with information objects. Information objects in this metaphor are collections of sign tokens. These can be considered as being data or operation (von Neumann principle of computer organization). Operations may be invoked on data to filter, project, reorder, reshape, and export or import data or views on them from or to other information spaces and to store them within such space. Furthermore customers may enter or leave information spaces.

Invocation of an information system supplied operation can be achieved by first identifying and locating it, positioning on the identified location and then launching it against the data to be processed. For more detail concerning information spaces, in particular definitions of the mentioned operation kinds see [21]. Information spaces may be subdivided into subspaces according to peculiarities of the business the information system creating the very space is about. This subdivision may be used to reduce the complexity of the navigation tasks that customers successfully need to perform when they operate an information system.

4 Quality driving development

The process of developing information systems must somehow be guided since lack of such guidance jeopardizes successful finishing the respective project. If figures given in the 1995 CHAOS report of the Standish Group, see www.standishgroup.com, are still valid then the respective threat needs to be taken serious. The guidance of course somehow must correlate with what the customer wants. However, it is well known, see, e.g. [8] that customers believe they recognize what they want once they meet it but cannot specify it in advance. Software represents knowledge ([1–3]) and successful information systems development thus is a learning process resulting in knowing what to implement and knowing how to do it. The evolutionary way in which the 'know what' is created implies that the customers requirements are changing and must be maintained. Respective so-called best practice has been incorporated into software processes such as the RUP, see [23] in the form of recommendation to manage requirements. The mentalistic point software development makes here is to assume that the inner parameter governing the customers' behavior comprises requirements and in particular quality expectations that customers change over time as they learn about their problem and how to solve it using an information system and that they use this inner parameter to decide about acceptance test and finally paying for the development effort.

Thalheim's co-design methodology (CDM) is an approach to information systems development that subdues the development process the information systems' mission statement. With respect to CDM information systems are conceptualized as a pyramid, known as abstraction layer model (ALM). For a brief overview of ALM and CDM see, e.g. [35] and figure 1. The

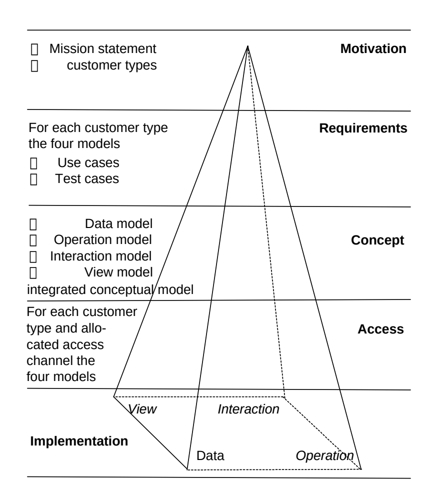


Fig. 1. Co-design methodology in abstraction layers

abstraction layers identified by ALM address motivation, requirements, conceptual model, access peculiarities and implementation. For all abstraction layers apart from the motivation layer CDM guides the developers to specifically deal with data, operation, interaction and views. If one considers programming language code as representation of executable models then one can say that CDM at all layers apart from the motivation layer asks developers to develop models of the data, operation, interaction and the views accessible to a particular customer type.

Feasibility of CDM and ALM for information systems design shows that a theory of information systems development can be based on semiosis and at the same time consider mentalistic aspects (such as purpose) as important. However, not considering mentalistic aspects makes it difficult to deal at all with what customers want and to direct the process of information systems development satisfactorily.

5 Quality in information systems use: models

The linguistic usage model of information systems implies that customers and information systems artifacts exchange linguistic expressions, i.e., composite signs. As was shown above this exchange can be understood as communication. The structural perspective definition of information systems introduces a purpose of using information systems for more efficiently doing daily business, solving problems and managing organizations. The way information systems are developed suggests that the communication the customer is involved in is a communication mediated by the information system and that it takes place between customers and experts. The communication partners can be separated by space, time, culture, occasion

or comparable. Knowledge and abilities of experts during information systems development are represented in the respective information system. The intermediation of this communication by the information system leads to the experts expertise being used more effectively and more economically. The task of successfully intermediating the communication between

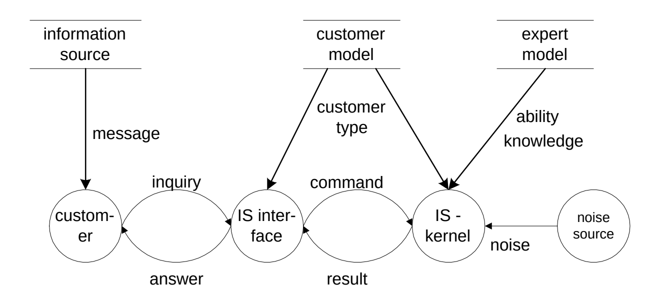


Fig. 2. Schema of inquiry processing

customers and users asks for a representation of a customer model being incorporated in the information system as well as a representation of an expert model. The main steps of this task are depicted in figure 2. In information systems development often the expert model is addressed as domain model or similar. Entity relationship modelling or object oriented analysis are standard approaches for obtaining the respective expert model. Some authors from a realistic point of view would argue that what here is called the expert model better would be called a real world model. This argument however in at least two respects is not satisfactory. Firstly, information systems as was addressed by several of the definitions above target organizations the reality of which essentially is socially constructed. Secondly, all our language achieved references to entities are references to already categorized and thus conceptualized entities. For more detail on this, see, e.g. the discussion of perceptual categorization in [13, pp. 102]. The purpose of the representation of the expert model in the information system is (of course) being able to generate replies to customer inquiries.

The necessity of representing a customer model in the information system is a consequence of two arguments. Firstly, different customers might find different answers to an issued (identical) inquiry helpful. Thus helping a large set of customers efficiently reaching their goal can reasonably only be achieved if information about these is built into the information system. Secondly, replying to customer inquiry in a consistent manner is necessary to allow the customer to easily use the reply. However, this requires information about the customer be built into the information system. It appears to be that explicitly representing customer models in information systems up to some degree can be neglected in case of traditional, i.e., non Web information systems since the customers of these systems often have no choice other than using the system. In Web information systems, however, this is different. They are open systems in so far as they offer their service to a more or less general audience. This openness

often is accompanied by competition, i.e., existence of systems with comparable service extent and quality. See [21] for more detail on open information systems.

At an initial level of sophistication customer types are dealt with in use case analysis. It was introduced to a wider audience by *Jacobson* in [19]. *Jacobson's* proposal now is part of UML (see [5]) as sub-language for use case modelling. It, however, usually is used only to deal with customers having qualitatively different functional requirements and is not based on a formal definition of the concept of customer type. The concept of customer type, however, can be applied in a much wider range of situations and allows also to address differences in non-functional requirements. Using it one can address the system usage as it differs between novices and experts, or men and women, or similar, see, e.g. [21] for more detail on customer types.

The linguistic usage model implies further the necessity of modelling during information systems development if modelling is understood as the process of working out customer model and expert model. It further also requires customers to model in that these need to associate imaginations to sign tokens exchanged with the information system artifact. Models now often are assumed to have characteristics similar to the ones given above, see [7, 26, 27, 29, 31, 36]. Both truncation and pragmatic property now imply the application of mentalistic concepts such as 'purpose' to understand the model concept and its application in information systems development. I count "purpose" as a mentalistic concept since it involves a perception of the world, an assessment of it as well as a decision to either maintain or change the actual state of the world. Purpose thus implies a particular structuring of the inner parameter governing the individual's behavior. I thus consider it as being outside the scope of Behaviorism.

6 Resume

We have discussed *Stamper's* dissenting position concerning some basic concepts proposed by the FRISCO concept as basic concepts for information systems. We have shown that a mentalistic position concerning the foundational problems of information systems is compatible with basing information systems theory and practice on semiosis. We further have argued that the linguistic usage model of information systems requires modeling and that discussing modeling needs to take into account mentalistic concepts such as purpose.

References

1. Philippe G. Armour. The Case for a New Business Model. *Communications of the ACM*, 43(8):19 – 22, August 2000.
2. Phillip G. Armour. The Fîve Orders of Ignorance. *Communications of the ACM*, 43(10):17–20, Oktober 2000.
3. Phillip G. Armour. The Laws of Software Process. *Communications of the ACM*, 44(1):15–17, Januar 2001.
4. John Longshaw Austin. *Zur Theorie der Sprechakte*. Philip Reclam jun., Stuttgart, 1979.
5. Simon Bernett, John Shelton, and Ken Lusin. *UML*. Schaum's outline series. McGraw-Hill, New York et al., 2001.
6. Peter Bernus and Günther Schmidt. Architecture of information systems. In Peter Bernus, Kai Mertins, and Günther Schmidt, editors, *Handbook on Architectures and Information Systems*, chapter 1, pages 1 – 9. Springer Verlag, Berlin et al., 1998.

7. Max Black. *Models and metaphors: studies in language and philosophy*. Cornell University Press, Ithaca, New York, 3rd. edition, 1966.
8. Barry Boehm. Requirements that Handle IKIWISI, COTS and Rapid Change. *IEEE Computer*, 33(7):99–102, Juli 2000.
9. Barry Boehm. Value-based software engineering. *ACM SIGSOFT Software Engineering Notes*, 28(2), March 2003.
10. Barry Boehm and Li Guo Hang. Value-based software engineering: Reinventing "earned value" monitoring and control. *ACM SIGSOFT Software Engineering Notes*, 28(2), March 2003.
11. John F. Cragan and Donald C. Shields. *Understanding Communication Theory: The Communicative Forces of Human Action*. Allyn & Bacon, Needham Heights, Massachusetts, 1998.
12. Stanislas Dehaene. *The Number Sense: How the Mind Creates Mathematics*. Penguin Books Ltd., London, England, 1999.
13. Gerald M. Edelman and Guilio Tononi. *Consciousness: how matter becomes imagination*. Penguin Group, 2000.
14. E. D. Falkenberg, W. Hesse, and A Olive, editors. *Information system concepts: towards a consolidation of views*. Chapmann & Hall, 1995. Proceedings IFIP WG8.1 Conferences ISCO3.
15. Eckhard D. Falkenberg, Wolfgang Hesse, Paul Lindgren, Björn E. Nilsson, Oei J. L. Han, Colette Rolland, Ronald K. Stamper, Frans J. M. Van Assche, Alexander A. Verrijn-Stewart, and Klaus Voss. *A Framework of Information System Concepts, The FRISCO Report (Web Edition)*. IFIP, ifip@ifip.or.at, 1998.
16. Richard L. Gregory, editor. *The Oxford companion to the mind*. Oxford University Press, London, New York, 1998.
17. Rudy Hirschheim, Heinz K. Klein, and Kalle Lyytinen. *Information Systems Development and Data Modeling, Conceptual and Philosophical Foundations*. Cambridge University Press, Cambridge, 1995.
18. Information technology —Software product evaluation — Part 1: General overview, 1999. International Standard ISO/IEC 14598-1.
19. I. Jacobson, M. Christerson, P. Jonsson, and G. Övergaard. *Object-Oriented Software Engineering-A Use Case Driven Approach*. Addison-Wesley Publishing Company, Wokingham, England et al., 1992.
20. Wilhelm Kamlah and Paul Lorenzen. *Logische Propädeutik : Vorschule des vernünftigen Redens*. Verlag J. B. Metzler, Stuttgart, Weimar, 1996.
21. Roland Kaschek, Wallace Catherine Schewe, Klaus-Dieter, and Claire Matthews. Story boarding for web-based information systems. 2003. Accepted for publication as book chapter.
22. Roland Kaschek and Sergiy Zlatkin. Where ontology affects information systems. In HeinrichC. Mayr, Michael. Godlevsky, and Stephen C. Liddle, editors, *ISTA 2003 Proceedings*, Bonn, Germany, 2003. GI.
23. Philippe Kruchten. *The Rational Unified Process: an introduction*. Object technology series. Addison-Wesley, Boston et al., 2nd. edition, 2000.
24. Peter C. Lockemann and Heinrich C. Mayr. *Rechnergestützte Informationssysteme*. Springer-Verlag, Berlin, Heidelberg et al., 1978.
25. Heinrich C. Mayr, Peter C. Lockemann, and Martin Bever. A Framework for Application Systems Engineering. *Information Systems*, 10(1):97 – 111, 1985.

26. Marvin Minsky. Matter, mind and models. In Marvin Minsky, editor, *Semantic information processing*, pages 425 – 432. MIT Press, Cambridge, Massachusetts and London, England, 1968.
27. Horst Oberquelle. On models and modeling in human-computer co-operation. In G. C. Van der Meer, M. J. Tauber, T. R. G. Green, and P. Gorny, editors, *Readings on cognitive ergonomics - mind and computers: proceedings of 2nd. European Conference*, pages 26 – 43, Berlin et al., 1984. Springer - Verlag Berlin Heidelberg.
28. William O'Grady, John Archibald, Mark Aronoff, and Janie Rees-Miller. *Contemporary Linguistics*. Bedford/St. Martin's, Boston, 4th. edition, 2001.
29. Edward S. Quade. Predicting the consequences: Models and modeling. In Hugh Miser and Edward S. Quade, editors, *Handbook of systems analysis: overview of uses, procedures, applications and practice*, pages 191 – 218. Elsevier Science Publishing Co., Inc., New York, 1985.
30. Arnim Regenbogen and Uwe Meyer. *Wörterbuch der philosophischen Begriffe*. Felix Meiner Verlag, Hamburg, 1998. begründet von Friedrich Kirchner und Carl Michaelis, fortgesetzt von Johannes Hoffmeister und von den Autoren vollständig neu herausgegeben.
31. Jeff Rothenberg. The nature of modeling. In Lawrence E. Widman, Kenneth A. Loparo, and Norman R. Nielson, editors, *Artifical intelligence, simulation, and modeling*, pages 75–92. John Wiley & Sons, Inc., New York et al., 1989.
32. Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer Verlag, Wien, New York, 1973.
33. Herbert Stachowiak. Erkenntnisstufen zum Systematischen Neopragmatismus und zur Allgemeinen Modelltheorie. In Herbert Stachowiak, editor, *Modelle-Konstruktionen der Wirklichkeit*, pages 87–146. Wilhelm Fink Verlag, München, 1983.
34. Herbert Stachowiak. Modell. In Helmut Seiffert and Gerard Radnitzky, editors, *Handlexikon Zur Wissenschaftstheorie*, pages 219–222. Deutscher Taschebuch Verlag GmbH & Co. KG, München, 1992.
35. Bernhard Thalheim. *Entity-Relationship Modeling*. Springer-Verlag, Berlin, Heidelberg, 2000.
36. Roelf Wieringa. *Algebraic foundations for dynamic conceptual models*. PhD thesis, Free University of Amsterdam, Amsterdam, The Netherlands, Mai 1990.

Perspektiven von Informatik und Wirtschaftsinformatik als technisch-politische Wissenschaften

Zielparameter für Informatik-Projekte zur Unterstützung von Vorgängen und Prozessen unserer Lebenswelt aus Sicht der Konstruktiven Wissenschaftstheorie

Ludger Eversmann

1 Einleitung

In Paul Lorenzens „Lehrbuch der konstruktiven Wissenschaftstheorie" heißt es in der Einleitung:

> „In der Wissenschaftstheorie beginnt sich die Einsicht, dass alle Wissenschaften (alle Theorien) nur aufgrund schon – teilweise – gelungener Praxis sinnvoll sind, in unserem Jahrhundert als sog. pragmatische Wende langsam durchzusetzen. Alle Theorien sind Redeinstrumente zur Stützung schon begonnener Praxis. Das ist für die *technische Praxis* allgemein anerkannt: physikalische Theorien stützen die vorwissenschaftliche Technik. So ist unsere Technik eine theoriegestützte Praxis geworden. Technik stellt aber immer nur die Mittel bereit für Zwecke. Über die Zwecke (bis zu den obersten Zwecken, den Lebensformen, die nicht mehr Mittel für anderes sind) muß in nicht-technischen Wissenschaften beraten werden. Eine vorwissenschaftliche *politische Praxis* der Gesetzgebung, die – zunächst innerstaatlich – den faktischen Pluralismus unverträglicher Lebensformen zu überwinden versucht, gibt es bei uns (im Rückgriff auf die klassische Antike – und trotz aller Rückfälle in bloße Machtpolitik) seit der Aufklärung. Das Ziel ist eine Pluralität verträglicher Lebensformen. Die Argumentationspraxis der Politiker ist für dieses „ethische" Ziel durch politische Wissenschaften als theoretische Instrumente zu stützen. Nur theoriegestütztes Argumentieren kann zu freiem (d.h. nicht erzwungenem) Konsens über die normative Ordnung unseres Zusammenlebens führen."[1]

Hieraus leitet sich offensichtlich u.a. auch ab, dass es in der Auffassung der konstruktiven Wissenschaftstheorie nicht „technisch-politische" Wissenschaften gibt, sehr wohl aber einen erklärten Anspruch an die technischen Wissenschaften, dass diese von ethisch-politischen Wissenschaften, also von theoriegestützten Beratungen und Argumentationen über das legitime und begründete „Wozu" der technischen Wissenschaften geleitet sein sollen. Das sprachlich-rationale, ethisch-politisch begründete „Wozu", die Reflexion auf als vernünftig anerkennbare „Zwecke" der technischen Wissenschaften spielt also in der Konstruktiven Wissen-

[1] Lorenzen (2000), S. 18

schaftstheorie eine besondere Rolle – eben dies sollte in diesem Beitrag mit der Wortwahl im Titel hervorgehoben werden.

Paul Lorenzen befindet sich mit seiner Definition des „Ethisch-Politischen" und der Konzeption der dialogischen Ethik der „Erlanger Schule" insgesamt nahe an den Leitideen der sog. Frankfurter Schule bzw. der Diskurs-Ethik und deren wissenschaftstheoretischer Fassung der normativen Bindung von Wissenschaft an Werte über das Konzept der Erkenntnisinteressen; er selbst sagt dazu: „Die politisch begründete Ethik führt in den Anwendungen (...) zu genau den selben Forderungen an das Argumentieren, wie die ‚Diskursethik' der Frankfurter Schule von Habermas und Apel. Die Unterschiede zur konstruktiven Wissenschaftstheorie der ‚Erlanger Schule' liegen nur in den Fundierungsproblemen. (...) Statt vom kommunikativen Interesse wird hier von der politischen Not posttraditionaler Staaten ausgegangen. (...) Die Forderung des konsensorientierten Argumentierens ist für beide Schulen dieselbe. Beide plädieren für den Primat der ethisch-politischen Vernunft – also (in der Formulierung von Habermas) gegen die *halbierte Vernunft* des technischen Denkens."[2]

D. h. also: in dieser – somit auch von der Frankfurter Schule vertretenen – Auffassung macht erst die ethisch-politische Vernunft aus dem „halbierten" zweckrationalen technisch-wissenschaftlichen Denken und Handeln die „ganze" Vernunft, erst der Primat der ethisch-politischen Vernunft leitet das technische Erkenntnisinteresse im Sinne der Etablierung einer aufgeklärten, kulturschöpfenden mitmenschlichen Praxis.

Diese doch nun sehr hervorgehobene Stellung der ethisch-politischen Vernunft in der Konstruktiven Wissenschaftstheorie scheint in einigen jüngeren Veröffentlichungen, die sich mit verschiedenen Aspekten des Themenkreises „Informatik und Konstruktive Wissenschaftstheorie" beschäftigen, keinen dieser Stellung entsprechenden Niederschlag zu finden. Vielleicht spiegelt sich hier auch eine anhaltende Vorliebe für ein naturwissenschaftlich-szientistisches Wissenschaftsverständnis wieder, das Überlegungen zur Normbegründung sowohl für die Anfänge („Prinzipien") als auch die Ziele oder Zwecke der Wissenschaften als „unwissenschaftlich – d.h. als methodisch nicht überprüfbar"[3] verwirft, oder dem subjektiven Belieben und Meinen überantworten möchte.[4] Vergegenwärtigt man sich einige Vorschläge zu Zielen oder „Visionen" für Informatik und Wirtschaftsinformatik, die in den Spalten einschlägiger Fachorgane in Umlauf sind oder auch in anerkannten Lehrbüchern vertreten werden, so stellt sich jedenfalls zumindest der Eindruck einer beachtlichen Heterogenität des vorhandenen Spektrums der „Meinungen" dazu ein; da wären etwa genannt

a) die „sinnhafte Vollautomation der Unternehmens" von Peter Mertens[5], neuerlich als „Vision" von W. König und A. Heinzl „bestätigt"[6];
b) die „Beseitigung sprachlicher Mängel", vielleicht auch „via chipbasierte Implantate" als curriculare Bildungsziele eine Informatik als „Sprachingenieurwesen", mit dem Ziel des „Fit-Machens" der Studenten für die „Cyberspace-Ära"[7]; oder
c) der Vorschlag eines „Langfristziels" für die Wirtschaftsinformatik als „Ausformung und Erprobung einer Theorie des Kollaborationsindividualisten oder Inividualkollaborateurs", mit der Erfolgsperspektive, „vielen Akteuren auf der Welt [zu helfen]

[2] Lorenzen (1991), S. 64
[3] Lorenzen (2000), S. 12
[4] vgl. etwa für die Wirtschaftsinformatik Müller-Merbach (2002), S. 300: „Beides, Wissenschaftstheorie und Wissenschaftsprogrammatik (...) steht gewissermaßen außerhalb der Wissenschaft selbst, und deren Ergebnisse entziehen sich weitgehend einer Beweisbarkeit. Vielmehr kommt Meinung zur Geltung, persönliche Überzeugung, Fürrichtighalten, Fürguthalten..."
[5] Mertens (95), S. 25 ff.
[6] König/Heinzl (2002), S. 508 - 511
[7] Ortner (2002a), vgl. dazu die Replik von Schefe (2002), sowie die Erwiderung von Ortner (2002b)

sich besser zu verhalten und weniger Fehler zu machen, was beispielsweise in gewisser Weise einem Ziel in der Pharmazie oder Medizin entspricht.“[8]

In einem Diskussionsbeitrag zur Fragestellung „Wie viel Wissenschaft(lichkeit) verträgt die Praxis?“ – eine Fragestellung, aus der aus konstruktivistischer Sicht sehr vieles zu sagen wäre – nennt Mertens als „Mission des Wissenschaftlers“ die „Suche nach Wahrheit“[9]. An diese sicherlich kaum strittige und in einiger Hinsicht verbindende oder auch verbindliche Feststellung anknüpfend soll nun zu zeigen sein, dass die Konstruktive Wissenschaftstheorie für Informatik und Wirtschaftsinformatik Vieles an orientierungsleitendem Klärungsvermögen zu bieten hat, dies sowohl für den Entwurf informations- und stoffverarbeitender maschineller Systeme als dem Gegenstandsbereich von Informatik- und Wirtschaftsinformatik im engeren Sinne, für den Entwurf sozio-technischer Arbeitssysteme, also von Organisationen, innerhalb derer mit maschineller Unterstützung Leistungen erstellt werden, als schließlich auch für den Gesellschaftsentwurf als dem übergeordneten normativen Rahmen lebensweltlicher Sinnstiftung und Perspektivbildung.

2 Konstruktive Wissenschaftstheorie

Paul Lorenzen hat in einem innerhalb seines Lebenswerkes spät entstandenen Text über „Philosophische Fundierungsprobleme einer Wirtschafts- und Unternehmensethik“[10] sehr knapp, klar und gewissermaßen kondensiert die wichtigsten Lehrinhalte der Konstruktiven Wissenschaftstheorie dargestellt, weshalb sie hier etwas ausführlicher rekapituliert werden sollen.

Lorenzen versteht Wissenschaftstheorie als Prinzipienlehre, die auf einem „kritischen Weg zwischen Dogmatik und Skepsis“, also einem kritisch nachvollziehbaren Weg die „Anfänge“, die Prinzipien der Fachwissenschaften nachträglich bzw. reflexiv, also „rückbeugend“ und rekonstruierend explizit „und damit lehrbar“ machen will. Die „kritische Philosophie“ in diesem Sinne, als Prinzipienlehre, „stellt sich die Aufgabe, auch die ersten Schritte und Zielsetzungen von Fachwissenschaften lehrbar zumachen.“ Auch wegen dieser Aufgabe der Reflexion über Zielsetzungen von Fachwissenschaften gehört für Lorenzen „Ethik als kritische Morallehre im gegenwärtigen Sprachgebrauch zur Wissenschaftstheorie.“ (S. 40/41)

Lorenzen erläutert dann in diesem Text knapp die „sprachkritische Wende in der Wissenschaftstheorie“ – die sich nach Begründung der modernen Logik seit Freges Begriffsschrift 1879 dann ab den 50er Jahren des letzten Jahrhunderts von Oxford ausgehend als „linguistic turn“ durchgesetzt hat, und dann zu dieser Aufgabenstellung der konstruktiven Wissenschaftstheorie geführt hat, „wissenschaftliche Fachsprachen zu konstruieren, als ob es sie noch nicht gäbe“. (S. 41) Er kommt anschließend zur praktizistischen Wende in der Wissenschaftstheorie, die auf der Einsicht fußt, dass in der Wissenschaftstheorie Konsens darüber erarbeitet worden sein muss, „*wozu* überhaupt Wissenschaften schrittweise begründet werden sollen“, bevor man „für die Fachwissenschaften interlingual verbindliche Sprachen (insbesondere syntaktische Kategorien und semantisch normierte Termini)“ konstruieren kann. (S. 42) Wie Lorenzen darstellt, wurde eine solche praktizistische Wende, „die hinter die traditionell herrschenden Kategorien religiösen oder philosophischen Denkens zurückgreift auf den vorwissenschaftlich lebenden Menschen, (wurde) *vor* dem ‚linguistic turn’ in der deutschen Prinzipienreflexion von Kant über Hegel, Feuerbach, Engels und Marx vollzogen. Seitdem ist die Forderung, Theorie ‚aus der Praxis für die Praxis’ zu begründen, in der deutschen Wissenschaftstheorie lebendig geblieben.“ Die Konstruktive Wissenschaftstheorie verbindet nun „die Forderung nach einer Begründung fachwissenschaftlicher Theorien als Stützen der – für das Leben, ja das Überleben nötigen – vorwissenschaftlichen Praxis“ mit dem konsequenten Einsatz der seit der sprachkritischen Wende zur Verfügung stehenden sprachkritischen Mittel. (S. 42)

[8] König/Heinzl (2002), S. 510
[9] Mertens (2003), S. 98
[10] Lorenzen (1991)

Lorenzen versteht folglich „physikalisch-technisches Wissen“ als Mittel zum Zweck der „Stützung vorwissenschaftlicher Praxis“: „Das Ziel jeder Technik, Effizienz von Mitteln für Zwecke, gehört zur technischen Praxis.“ Im Laufe der Kulturgeschichte bildete sich nun z. B. die „Wirtschaft als organisierte Technik“ heraus. „Hochkulturen sind durch hochvermittelte Lebensvollzüge definiert. Diese erfordern *Arbeitsteilung* und damit eine soziale *Organisation* der Mittelbeschaffung: ein Wirtschaftssystem.“ Jedoch bleiben physikalisch-technisches Wissen und technische Vernunft immer als Mittel einem „letzten“ Zweck zugeordnet: „Wirtschaften bleibt stets ein organisiertes technisches Handeln, das die Mittel für Lebensvollzüge als obersten Zweck beschafft.“ (S. 44)

Diese „Lebensvollzüge als oberste Zwecke“ sind nun aber, über eine lediglich „technisch-ökonomische Begrifflichkeit“ hinaus, zu bestimmen. „Denn, wenn man seinen Blick auf das Technisch-Ökonomische beschränkt, auf Arbeit und Verbrauch, dann entsteht überhaupt kein moralisches Problem: der pausenlose Kampf, das marktwirtschaftliche ‚Spiel der Kräfte’, ist moralfrei wie das Leben der Tiere.“ (S. 45)

Lorenzen schränkt folgend diese etwas drastische Formulierung ein (von ihm angenommenen ‚Leserprotesten’ zuvorkommend), indem er darauf verweist, dass „die freie Marktwirtschaft als Organisation (...) [nicht] eine [biologische] Instinktleistung, sondern eine eigene Kulturleistung“ sei. Er verwahrt er sich aber dagegen, Kultur „biologisch“ bzw. darwinistisch als „struggle for life“ zu verstehen, und setzt sein Verständnis des Menschen „als politisches Lebewesen“ dagegen. Lorenzen fragt nun, *inwiefern* der Mensch ein „politisches Lebewesen“ ist, was ihn als Kulturwesen über den biologisch-darwinistischen Kampf ums Dasein erhebt, was den Menschen denn also von seinem biologischen Vorfahren, vom Affen unterscheide. Und hier, so Lorenzen, müsse „eine andere Antwort gefunden werden als die, „dass er ein technisch-effizienter Raubaffe sei. Dass er in den Demokratien nur noch mit wirtschaftlichen Mitteln kämpft, genügt auch nicht.“ (S. 45–48)

Die Antworten findet Lorenzen „..in der Philosophiegeschichte. Besonders in der deutschen Tradition (seit der Aufklärung mit Kant und Hegel) wird die Entwicklung des Menschen über den Affen hinaus nicht primär in der technisch-ökonomischen Effizienz (gestützt durch die Machtmittel der Staatsorgane) gesehen, sondern in einer freien Kultur der Geister. Wirtschaft und Machtpolitik sind nur ‚zivilisatorische Vorbedingungen geistiger Kultur in der dreifachen Gestalt von Kunst, Religion und freier Wissenschaft. Erst hier erhebt sich der Mensch zu dem dreifaltigen Ideal des Schönen (Kunst), des Guten (Religion) und des Wahren (freie Wissenschaft).“ Wie Lorenzen weiter erläutert, sei diese „Hegelsche Dreifaltigkeit“ eines „absoluten Geistes“ aus Kunst, Religion und freier Wissenschaft eine „christliche Umdeutung der griechischen Aufklärung“; um diese Umdeutung rückgängig zu machen, müsse an die Stelle der „Religion“ das „Ethisch-Politische“ gesetzt werden, „und das ‚Gute’ ist durch ‚Gerechtigkeit’ zu ersetzen.“ (S. 48/49)

Den „Terminus“ „gerecht“ definiert Lorenzen nun so: „Ein Gesetz heißt gerecht, wenn es die allgemeine, freie Zustimmung der Bürger findet.“ (S. 54) Das Bindestrich-Wort „ethisch-politisch“ soll „die Einheit einer Moral (als Gesinnung und Haltung) mit einer an Gerechtigkeit statt an Macht orientierten Politik“ als „Pointe der platonisch-aristotelischen Geisteswissenschaften (sie tragen traditionell den obsolet gewordenen Namen ‚praktische Philosophie’)“ zum Ausdruck bringen. (S. 50)

Das Ideal des herrschaftsfreien Diskurses bzw. des (platonischen) Dialoges als herrschaftsfreie, symmetrische und transsubjektive Argumentationen über Normen, Gesetze, Richtlinien etc. verweisen also in eine normativ-verbindliche Ausgestaltung sowohl zwischenmenschlicher Verhaltensweisen als auch politischer Praxis; diesem „Herstellungs-“ bzw. Erkenntnisinteresse wird nun also auch Wissenschaft zugeordnet bzw. verpflichtet.

3 Was konstituiert die Informatik als Wissenschaft?

Die Konstruktive Wissenschaftstheorie will also reflexiv, „rückbeugend“, die Anfänge der Fachwissenschaften explizit und damit lehrbar machen. Eine Reflexion auf die Anfänge der Informatik in diesem Sinne hat Peter Janich 1993 vorgelegt[11]. P. Janich hat seinen Beitrag beschrieben als „Skizze (..), wie der Informatik dazu verholfen werden kann, sich zur Wissenschaft zu konstituieren: es geht darum, heute bereits technisch verfügbare Leistungen von Maschinen dadurch zu begreifen, dass sie in einen Konstitutionszusammenhang von vor- und außerwissenschaftlichen lebensweltlichen Praxen nach Zweck und Mittel begriffen werden.“ (a.a.O., S. 67)

Verfolgt man nun die Entwicklungslinien der Informatik bis zurück zu Alan Turings „*a*-Maschine“[12], die zu nichts anderem bestimmt ist, als berechenbare Zahlen von nicht-berechenbaren zu unterscheiden, so findet sich hier aber offenbar nicht eine vorwissenschaftliche lebensweltliche Praxis vor, die – wie etwa eingeübte Praxen des Abzählens und des Umgangs mit Quantitäten von Gegenständen als Konstitutionszusammenhang der Arithmetik – als Anfang der Informatik zu begreifen wäre. Es findet sich hier auch nicht der Begriff „Information“ oder Informationsverarbeitung, sondern nur ein – lesender oder schreibender – Umgang mit Symbolen. Typisch und wesentlich für die von Turing beschriebene „*a*-Maschine“ bzw. die Universale Turingmaschine ist, dass sie „Probleme lösen“[13] kann, sofern diese effektiv berechenbar sind, also durch Berechnung in endlicher Zeit gelöst werden können. Auch „Berechnung“ findet sich hier eigentlich nicht als ausschließlich auf Zahlensymbole anzuwendende Operation, sondern als Exempel vollständig determinierter Abfolgen von Bewegungen einer „Maschine“. Ist die Maschine mit den notwendigen Sensoren und Effektoren ausgestattet, um mit stofflichen Objekten hantieren zu können, so kann sie eben auch Stoff „verarbeiten“ oder bearbeiten – wie der Entwicklungsstand der Robotik zeigt, tut sie dies inzwischen recht erfolgreich. In der ursprünglichen „Idee“ der Turing-Maschine findet sich also weder die Informationsverarbeitung, noch die Substitution menschlicher Informationsverarbeitung durch diese Maschine, noch die Stoffverarbeitung, noch auch die Substitution menschlicher Stoffverarbeitung durch die Maschine – maschinelle Informations- und Stoffverarbeitung, und dies als Substitution menschlicher „Ver-Arbeitung“, also als Entlastung arbeitender Menschen, sind wohl erst entstanden, als die Turing-Maschine als „Computer“, als technisches Arbeitsgerät, in einem minimalen Reifegrad bereits vorhanden war; und die „Informationsverarbeitung“ entstand wohl vor allem deshalb geschichtlich vor der Stoffverarbeitung, weil die maschinelle Operation mit als Information zu interpretierenden Symbolen technisch leichter zu realisieren war als ein universal programmierbarer maschineller Umgang mit Stoffen oder Körpern.

Worin besteht dann also „das Handeln des Informatikers“? Wie ist es „als Mittel für erkennbare, erkannte und anerkennenswerte, d. h. legitimierbare Zwecke“[14] begreiflich zu machen? Ist die Entwicklung stoff- und informationsverarbeitender Maschinen, die Entwicklung von „Informationssystemen“ oder Kommunikationssystemen, von „Arbeitssystemen“, von Produktionssystemen, die die Effizienz menschlicher Arbeit erweitern und verbessern, die den Menschen insgesamt von berechenbarer Arbeit entlasten, ein anerkennenswerter und legitimierbarer Zweck? Was wären dann die langfristigen Perspektiven der Informatik? Bedarf der Zweck „maschinelle Substitution menschlicher berechenbarer Arbeit“ einer expliziten Legitimation?

Nach der hier vertretenen Auffassung ganz offensichtlich ja, und wie noch ausführlicher entwickelt werden soll, verlangt gerade der Blick auf die Wirkungsentfaltung der entwickelten Informations- und Automationstechnik im Rahmen der sich beschleunigenden Wirtschaftsevolution eine sehr sorgfältige Evaluation der projektierten Eingriffe in die Lebenswirklichkeit der Menschen.

[11] Janich(1993)
[12] Turing (1987), S. 21
[13] „Lösbare und unlösbare Probleme“, Turing (1987), S. 63 ff.
[14] Janich (1993), S. 68

Wird an dieser Stelle aber einmal unterstellt, dass (auch) die Entwicklung von Informationssystemen zum legitimen Gegenstandsbereich der Informatik gehört, dann kann hier nun in aller Kürze der Blick gerichtet werden auf den konstruktivistischen Ansatz des Systementwurfs; E. Ortner etwa hat diesen Ansatz in Abgrenzung gegen einen „empiristischen“ und einen „formalistischen“ Standpunkt erläutert[15]. Wie Ortner darlegt, werden in konstruktivistischer Vorgehensweise und Methodik „sowohl das Mittel (z. B. ein Workflow-Modell oder das Relationenmodell) als auch der Gegenstand (z. B. das Beschaffungs- oder Vertriebswesen eines Unternehmens), auf den das Mittel angewendet wird, sprachkritisch (re)konstruiert..“ (S. 39) Zur Charakterisierung der konstruktivistischen Vorgehensweise sagt Ortner: „Die faktische Genese einer Sprache wird bei einem konstruktivistischen Ansatz durch eine kritisch-rationale (normative) Genese ersetzt.“ Die Mittel oder Gegenstände, „die zur Debatte stehen, [werden] (..) noch einmal rational (ab ovo) – mit der Möglichkeit, bestehende Verhältnisse (Sachverhalte) normativ zu verändern – rekonstruiert.“ (S. 40)

Dass die Entwicklung einer normierten Terminologie als für die „Fachwissenschaften interlingual verbindliche Sprachen (insbesondere syntaktische Kategorien und semantisch normierte Termini)“ sowie eben auch als (Fach-)Semantik für den „lebensweltlichen“ Anwendungsbereich einer Software-Applikation einerseits wünschenswert, andererseits auch möglich ist, wäre also – s. o. – mit Verweis auf Lorenzen einerseits sowie anderseits sicher auch auf die Existenz in betrieblicher Praxis bewährter standardisierter und daher auch normierter Anwendungssoftware zu stützen. Solche Software enthält zwangsläufig auch eine anwendungsgebietsspezifische Terminologie, und diese ist *idealerweise* möglichst verallgemeinerungsfähig angelegt, mit Lorenzen also: rational und normativ (re)konstruiert. Auch wäre etwa die Scheersche Wirtschaftsinformatik als Entwicklung von „Referenzmodellen für industrielle Geschäftsprozesse“ ja im Grunde kaum anders denkbar, als dass sowohl Strukturen des Anwendungsbereiches (i.w. Industriebetriebe) als auch das fachspezifische Sprechen über diese Strukturen und „Prozesse“ einer rational rekonstruierenden Normierung zugänglich sind. Im Hause SAP sind, wie man sich erinnern wird, während der 1990er Jahre nicht unerhebliche Entwicklungsaufwendungen getrieben worden, um ein derartiges Meta-Informationssystem bzw. Repository mit „Ontologie“ bzw. fachlicher Terminologie wie von Ortner beschrieben in die Software zu integrieren.

Im Ergebnis wäre demnach die Aufgabe des (Wirtschafts-)Informatikers also durchaus auch, gewissermaßen die Innensicht eines „verallgemeinerten“ Anwenders zu übernehmen und zu entwickeln, um so schließlich auch – sicherlich im engen Kontakt mit den Anwendern – material-analytische Normierungen zu leisten; möglicherweise auch in einem „hypothetischen“ Kontakt mit einem „ideellen“, verallgemeinerbaren Anwender, und der hypothetischen Berücksichtigung seiner als verallgemeinerungsfähig, auch als gerechtfertigt zu unterstellenden fachlichen Interessen und Arbeitsziele. [16]

4 Wie notwendig ist Wissen über Ziele? Die Konstruktivität der Informatik

P. Janich führt den Gedankengang einer Darstellung des „modernen Konstruktivismus“ im Kontext mit wissenschaftstheoretischen Fragestellungen für die Wirtschaftsinformatik[17] hin

[15] Ortner (2002a)
[16] „Eine der grundlegenden Aufgaben der Wirtschaftsinformatik besteht darin, das aus den Anwendungsbereichen stammende Fachwissen zu rekonstruieren, dass es a) mit Hilfe des Computers verarbeitet und b) von den Anwendern zur Erreichung ihrer Ziele effizient genutzt werden kann.“ Ortner (1998), S. V 13
[17] „Moderner Konstruktivismus: methodisch-kulturalistisch“. Vortrag von P. Janich anlässlich der Tagung „Wirtschaftsinformatik und Wissenschaftstheorie“ im Oktober 1998 an der Universität GH Essen

zu der „Folgerung – Wirtschaftsinformatik – wozu?“ – also zur Reflexion auf das Wissenschaftsziel. Wäre nun etwa mit der Aufgabe der Rekonstruktion von Fachwissen für diverse Anwendungsbereiche und dessen Implementation in Informationssystemen zur effizienten Arbeitsunterstützung die Frage „Wirtschaftsinformatik – wozu?“ vollständig und zufriedenstellend beantwortet?

Die Fragen nach Zielen, Gegenstand, Inhalt, Begriff oder „Wurzel“ einer Wissenschaft sind offensichtlich eng miteinander verknüpft; eine treffende intensionale Definition z.B. sollte die Ableitung einer vollständigen extensionalen Definition ermöglichen, und auf der Grundlage der erfassten Extension eines Faches, also seiner Teilbereiche, eben auch seiner Geschichte oder seiner geschichtlichen Wurzeln, sollte die Angabe eines „Super-Zieles“ beruhen. Die – viel und kontrovers diskutierte – ACM-Definition von 1989: „Die grundlegende Fragestellung der Informatik ist ‘Was kann effizient automatisiert werden?’“[18] etwa mag in diesem Sinne eine zutreffende Auskunft darüber geben, welche Fragestellung bzw. welches Gestaltungsziel für die Informatik grundlegend ist. Die Konstruktive Wissenschaftstheorie stellt aber darüber hinaus die Frage nach der Rechtfertigung, nach den *legitimierbaren* „Ober-Zwecken“ einer Wissenschaft, und da scheint eine Antwort wesentlich schwieriger zu geben, insbesondere, wenn – wie dies der Zielvorschlag der ‚sinnhaften Vollautomation des Unternehmens’ implizit ja tut – ein Überschreiten maschinell unterstützter Arbeitsproduktivität über die Grenzen hinaus unterstellt wird, die die generelle Möglichkeit menschlicher existenzsichernder Erwerbsbeschäftigung in der Wirtschaft prinzipiell noch bestehen lassen würden. In einem Universum mit ausschließlich mikro-ökonomisch hergeleiteten Wert-Parametern mag das Motiv „Verbesserung der Wettbewerbsfähigkeit des Unternehmens durch Lohnkostenminimierung“ eine so unbezweifelbare Gültigkeit besitzen, dass die „visionäre“ Vorstellung einer Komplettierung denkbarer maschineller Substitution menschlicher Arbeit bis hin zur „Voll-Automation“ aus bloßen Kostengründen schon fraglos gerechtfertigt erscheint. Aber aus der Perspektive einer Wissenschaftstheorie, die ihren „Sitz im Leben“ der Menschen hat und in einer den Lebensinteressen der Menschen verantwortlichen Weise ihre Wirkungsintentionen zu rechtfertigen und zu formulieren hat, fangen mit einer solchen „Vision“ von Gestaltungsziel die Fragestellungen doch erst an; insbesondere dann, wenn einmal das „technische“ Vermögen als gestalterisches Potenzial, menschliche Arbeit tatsächlich sehr umfassend auf berechenbare Automaten zu übertragen, als gegeben unterstellt wird. Dürfen wir dann, was wir können? Unter welchen Bedingungen und Voraussetzungen? Hier ist nun ethisch-politisches Orientierungswissen offensichtlich unverzichtbar, und hier bietet die Konstruktive Wissenschaftstheorie mit ihrer Anbindung an das sinnstiftende Kulturgut der Aufklärung eine überpositive, über bloße Macht- oder Meinungswillkür oder –beliebigkeit sich erhebende Autorität, Diskurse über normative Zielorientierungen zu konstituieren und auf diese Weise darüber zu befinden.

5 Informatik im lebensweltlichen Umfeld der Organisation der Mittelbeschaffung

In der Ökonomie – zumindest zu Teilen – oder auch in Soziologie, Sozial- oder Politikwissenschaften, sicher auch in der Informatik, ist der hiermit umrissene Problemkreis inzwischen seit Jahrzehnten in der Debatte, und seit Erscheinen der ersten Publikationen zum Thema „Zukunft der Arbeit“ bzw. zum „Ende der Arbeitsgesellschaft“[19] auch in einer breiteren Öffentlichkeit. Im wesentlichen wird hier eine Entwicklung einer sich scherenartig erweiternden Divergenz zwischen technologisch – also durch in der Wirtschaft angewandte Informatik – erweiterten Produktionsmöglichkeiten und durch relative Sättigung und stagnative Tendenzen hervorgerufenem Nachlassen der Produktionsnachfrage gesehen. Darüber hinaus argumentiert in jüngerer Zeit etwa der Ökonom Amartya Sen, immerhin Nobelpreisträger für Wirtschaftswissenschaften d. J. 1998, dass einer – diese Problematik ja entschärfenden – Perspektive infiniten Wirtschafts- und Güterwachstums in aller Zukunft auch der legitimierende Sinn fehle; vielmehr sieht er Entwicklung als eine „Perspektive der Freiheit“, als Zuwachs an „Freiheiten bei der Wahl der Lebensführung“: „Tatsächlich haben wir im allgemeinen hervorragende Gründe, uns mehr Einkommen und Reichtum zu wünschen. Doch nicht, weil Einkommen und Reichtum um ihrer selbst willen erstrebenswert sind, sondern weil sie in der Regel wunderbare Allzweckmittel sind, um eine größere Freiheit bei der Wahl der von uns als vernünftig eingeschätzten Lebensführung zu gewinnen.“[20]

Einer der prominentesten Wachstumspessimisten ist sicherlich John M. Keynes, der seine Annahme eines Einmündens der Wirtschaftentwicklung in eine stagnative Phase im wesentlichen mit der Beobachtung begründete, dass in einer „reichen“ bzw. reifen Wirtschaft mit steigendem Einkommen immer kleinere prozentuale Einkommensanteile in rein konsumtive Verwendungen gelangen, sondern gespart werden. Wie der Ökonom J. Stiglitz – ebenfalls ein Nobelpreisträger – jüngst darlegte[21], wird oder wurde dieses Argument auch von Vertretern einer „angebotsorientierten“ neo-liberalen Volkswirtschaftslehre gestützt, die es jedoch verwenden, um damit eine Notwendigkeit von Einkommensspreizungen und „Ungleichheit“ zu begründen, denn die höheren Sparquoten der hohen Einkommen fördern nach ihrer Auffassung die Kapitalakkumulation und damit die Investitionstätigkeit.

Es war nun aber in den vergangen Jahren sehr deutlich zu beobachten, dass zwar enorme Mengen freier Liquidität sowohl bei den Unternehmen als auch in privatem Sparvermögen vorhanden waren, dass diese aber offensichtlich keineswegs in produktive und arbeitsplatzschaffende Investitionen gelangt sind, sondern in häufig rein spekulative Verwendungen, also spekulative Aktien- oder Währungstransaktionen an den Börsen und Devisenmärkten, mit dem nun zu beklagenden Ergebnis, dass enorme Mengen von geschaffenen Werten wirtschaftlich vollkommen wirkungslos vernichtet worden sind; diese aktuell zu beobachtenden Entwicklungen scheinen somit eher Keynes’ Wachstumspessimismus zu stützen.

Keynes’ Auffassung war also die, dass Sättigung in die Stagnation führe: „Je reifer eine Gesellschaft und je reicher ihre Kapitalausstattung, desto geringer die erwartete Rendite zusätzlicher Investitionen und desto schwächer folglich auch die Investitionsnachfrage; also weitet sich die Lücke der effektiven Gesamtnachfrage aus. Das ist die Situation, in der die Tugend des Sparens – gesamtwirtschaftlich gewendet – sich als Fluch erweist, nämlich als Auslöser/Verstärker von Nachfrageschwäche und Krise.“[22] Vergegenwärtigt man sich hierzu den derzeitigen Stand der internationalen Leitzinsen von z. B. 1,25% in den USA, 0,1% in Japan und seit neuestem 2,0% in der Euro-Zone, so scheint sich doch zu bestätigen, dass Kapital inzwischen nahezu kostenlos zu haben ist, dennoch aber keineswegs in erwartbarem Umfang investiert wird.

6 Ziel-Parameter für zukünftige Informatik-Projekte

Man wird also – ohne diesen Gedanken hier erschöpfend oder gar zwingend ausführen zu können – folgern dürfen, dass wachstumsskeptische Annahmen doch immerhin einige Plausibilität für sich haben. Wenn nun aus plausiblen Gründen stagnative Tendenzen der Wirtschaftsentwicklung aus endogenen Gründen (also wegen zunehmend gesättigter Märkte aufgrund einer reichen Güterausstattung der Haushalte) angenommen werden können bzw. müssen, und wenn eine wie etwa von der ACM formulierte „General-Intention *effiziente Automation*“ für Informatik und Wirtschaftsinformatik nicht als vollständig gegenstandslos von der Hand gewiesen werden kann, dann stellt sich der Informatik und der Wirtschaftsinformatik

[18] Denning et al. (1989), aus Coy (1992), S. 3
[19] vgl. etwa Ralf Dahrendorf: „Die Arbeitsgesellschaft ist am Ende“, in „Die ZEIT“ vom 26. Nov. 1982

[20] Sen (2002)
[21] Stiglitz (2002)
[22] Willke (2002)

gerechter- und billigerweise in weiterer Konsequenz folgende zu beantwortende Fragestellung, und zwar als recht anspruchsvolles theoretisches Problem:

> *Wenn – offensichtlich kontrafaktisch – unbegrenzte maschinelle Rechenleistung, unbegrenzte Rohstoffe und unbegrenzte Energien unterstellt werden: sind Prinzipien benennbar, die eine rationale, geordnete Gestaltung der sozialen Organisation der Mittelbeschaffung wie auch der Gesellschaft für diese Unterstellung gewährleisten?*

Diese Fragestellung hat offensichtlich bisher keine systematisch entwickelte Beantwortung gefunden. Mertens hat lediglich diese von der ACM identifizierte General-Intention auch für die Wirtschaftsinformatik ausgemacht und als solche benannt, bzw. explizit auf den Anwendungsbereich Industrieunternehmen konzentriert, ohne jedoch eine solche Zielsetzung auch nur annähernd hinreichend zu begründen, oder sie, wie es kulturalistisch notwendig ist, auf ihre gesellschaftlich-ökonomischen bzw. eben lebensweltlichen Konsequenzen hin zu untersuchen. Im übrigen gibt es vielleicht einige kursorische Bemerkungen dazu („kindische Vorstellung eines vollautomatisierten Schlaraffenlandes“), oder „utopische“ Zukunftsentwürfe.

D. h. also es kann kaum hinreichend sein, innerhalb eines kurzfristigen und partikularen Orientierungs- bzw. Interessenhorizonts für das Gelingen des jeweils nächsten Informatik-Projektes besorgt zu sein (wobei als Erfolgsmaßstab zumindest für Wirtschaftsinformatik-Projekte ein nachweisbarer Return on Investment kaum zu schlagen sein wird), diesen Trend der erfolgreichen Substitution menschlicher Verwaltungs- und Steuerungsarbeiten durch programmierbare informationsverarbeitende Maschinen auch auf stoffverarbeitende Maschinen zu generalisieren[23] und dann hell am Horizont die „Vision“ vollautomatisierter Unternehmen aufscheinen zu sehen. Es bedarf keines besonderen ökonomischen Sachverstandes um zu verstehen, dass eine Volkswirtschaft mit lauter vollautomatisierten Unternehmen keinen Wohlstand entfalten könnte – aus vielerlei Gründen, von denen einer der ist, dass ein volkswirtschaftlicher Güter- und Leistungskreislauf dann eben unterbrochen wäre bzw. nicht existierte. Im denkmöglichen Extremum stünden einige oder nur noch ein „Unternehmer“ – im Besitz theoretisch unendlicher Produktionsmöglichkeiten – einer Masse von potenziellen Konsumenten gegenüber, die mangels Beschäftigung keinerlei Kaufkraft besitzen; Produzent(en) und Konsumenten könnten also keinerlei wohlstandserweiternde Kooperation eingehen.

Mit Bezug auf die Frage nach Forschungsstrategien ergäbe sich also die gewissermaßen geschichts-spezielle Aufgabenstellung, Prinzipien der Organisation der wirtschaftlichen Mittelbeschaffung, der Güter- und Faktorallokation, der Produktionsorganisation etc. zu entwickeln, die sozusagen Automaten-kompatibel sind, die also auch dann wohlstandserhaltend oder –erweiternd operieren können, wenn maschinelle Nutzkapazitäten in denkbar unbegrenztem Umfang zur Verfügung stehen.

Allgemeiner wäre im Einklang mit den Leitideen der Konstruktiven Wissenschaftstheorie zur Entwicklung einer Forschungsstrategie zu sagen, dass es jeweils darum gehen müsste, a) in theoriegestützten Beratungen materielle Ziele zu identifizieren, die als würdig anerkannt werden können, wissenschaftliche Forschungskapazitäten zu binden, und b) dann die methodisch geeigneten bzw. Informatik-spezifischen technischen Mittel zu identifizieren, die geeignet erscheinen, diese Ziele optimal zu unterstützen. Zur Fortführung der in diesem Beitrag

[23] H. Jungclaussen entwickelt eine „Architektur kausaldiskreter Systeme“ und macht deutlich, dass sich der Steuermechanismus eines automatischen Fertigungssystems nur in einem einzigen Punkt von dem eines informationsverarbeitenden Systems unterscheidet: ein Prozessor, der die Aufgabe eines Steueroperators übernimmt, hat im Falle von Informationsverarbeitung die Aufgabe, den nächsten Befehl zu holen, Steuersignale zu generieren und die befohlene Operation auszuführen; im Falle der Steuerung eines Fertigungsprozesses entfällt dagegen die Aufgabe der Ausführung, da sie von einem NC-Maschinen-Netz ausgeführt wird. „Dies ist der einzige Punkt, in dem sich der Steuermechanismus eines automatischen Fertigungssystems von dem eines IV-Systems unterscheidet.“ Jungclaussen (2002), S. 459

angerissenen Problematik könnten m. E. folgende thematische Schwerpunkte zu einer Forschungsstrategie gehören:

1) weitere Bearbeitung der Fragestellung: „What Computers can’t do“ bzw. „What Computers can do“; also 1-a): was werden Automaten – unabhängig von erreichbarer Rechenleistung – niemals „tun” können, verglichen mit menschlichen Handlungskompetenzen und mentalem Vermögen, und 1-b): was *sollten* sie tun können, um evtl. vorstellbare zukünftig erreichbare Rechenleistungen optimal ausschöpfen zu können?[24] Auf dem Wege der Bearbeitung dieser Fragestellung – auch mit philosophischen Erkenntnismitteln – können z. B. Aussagen darüber gewonnen werden, welche Aufgaben und Tätigkeiten prinzipiell menschlicher Handlungs- und Verantwortungskompetenz vorbehalten bleiben müssen, und welche anderen, als effektive Verfahren beschreibbaren Aufgaben optimal maschinell unterstützt bzw. ausgeführt werden können.
2) weitere Bearbeitung der ökonomischen „Bestandsaufnahme“: als wie signifikant sind Indikatoren stagnativer Tendenzen wirtschaftlichen (Güter-)Wachstums einzuschätzen? Ist hier eine Klärung der Diskussionslage zu erwarten bzw. zu erreichen? Gibt es andererseits Strategien, Forschungsstrategien der Informatik von der Frage des wirtschaftlichen Wachstumsverlaufs zu entkoppeln, die also für beide alternativ anzunehmenden Wachstumsverläufe als „kulturell wertvoll“ einzuschätzen sind?
3) Bearbeitung der Frage: wo liegen neuartige und zusätzliche ökonomische bzw. kulturelle Wertschöpfungspotenziale, die unter zielführendem Einsatz von Informations- und Automationstechnik zu erschließen sind?

Es ist hier daran zu erinnern, dass gegenwärtig in zunehmendem Maße Sättigungserscheinungen (auch) für die Märkte der Informationstechnik diagnostiziert werden.[25]

Wege zu neuer bzw. zusätzlicher Wertschöpfung werden nun - jedenfalls von dessen führenden Protagonisten - in dem produktionswissenschaftlichen Konzept der „Mass Customization“ gesehen, das deshalb an dieser Stelle erwähnt werden soll; es wird vorgestellt als informationstechnisch ermöglichtes Konzept von Güterproduktion unter Umfeldbedingungen weitgehender Sättigung der (Welt-) Märkte und globalen Verdrängungswettbewerbs der Unternehmen[26] und ermöglicht zusätzliche und neuartige Wertschöpfung dadurch, dass einem Verbraucher individuell (möglichst) maßgefertigte Produkte zu den geringen Kosten von Massenware angeboten werden. Es sei hier – ohne dass an dieser Stelle ausführliche Erläuterungen möglich wären – berichtet, dass das hier verfolgte Prinzip hochgradig produktiver und simultan hochgradig *flexibler* Produktion nach Auffassung des Verfassers im Keim auch den Ansatz einer Lösung des oben andeuteten Problems der gesellschaftlichen Güterversorgung unter Bedingungen nahezu unbegrenzter maschineller Arbeitsunterstützung enthält.[27]

Diese „Mass Customization“ stellt gegenwärtig eine Herausforderung dar für praktisch das gesamte Spektrum der IuK-Technologien, darunter auch die Künstliche Intelligenz mit Wissensrepräsentation und Wissensverarbeitung; zu bewältigende Aufgaben sind hier etwa die

[24] vgl. etwa Walthelm (2003)

[25] aktuell z. B. in der Computer-Zeitung Nr. 24 / 10. Juni 2003, S. 6: „Die IT verliert den Nimbus des Besonderen“; zitiert wird etwa ein Artikel der Harvard Business Review (HBR) mit der „Meinung, alles, was die Unternehmen heute an IT benötigen, sei bereits vorhanden, wenn auch nicht immer optimal genutzt“, sowie die „Bedenken über den Bedarf der Unternehmen“ des SUN-Chefentwicklers und –Mitbegründers Bill Joy auf dem World-Economic-Forum in Davos: „Was ist eigentlich, wenn die Unternehmen schon alles haben?“ Ein Artikel mit ähnlichem Tenor erschien in der Computerwoche Nr. 3 vom 17. Januar 2003: „Mit neuer Wertschöpfung aus der Krise. Nach Jahren ungebrochenen Wachstums befindet sich die IT-Branche erstmals in einer nachhaltigen Rezession. Die Märkte sind gesättigt....“ (S. 36/37)

[26] vgl. insbesondere Piller (2000)

[27] Den Versuch einer ausführlichen Begründung hat der Verfasser unternommen in: Eversmann (2003a). Einen Überblick über die hier vorgelegte Argumentation gibt Eversmann (2003b).

Visualisierung dreidimensionaler Körper (u. U. in Bewegung) mit verschiedenen Oberflächenstrukturen als Entwurfsmuster für Produktideen (sog. Konfiguratoren), die Generierung von CAD-Entwürfen, Stücklisten, Arbeitsplänen etc. aus solchen Produktentwürfen, die Verfeinerung der maschinellen Produktionsplanung und -steuerung mit Maschinenbelegungsplanung sowie auch Roboter-Einsatzplanung und -steuerung; genannt seien weiter etwa die Integration von Internet-Technologien und ERP-Software über objektorientierte Ansätze, also die B2C- als auch B2B-Integration, die Integration technischer Produktionssteuerung mit kaufmännischen Steuerungsinformationen, die maschinelle Übersetzung von CAD-/CAM-Daten in NC-Maschinenprogramme etc.

Die Steigerung technisch-ökonomischer Effizienz dürfte per Saldo wohl auch aus konstruktiver bzw. methodisch-kulturalistischer Perspektive einen Erfolgsgradmesser künftiger Informatik-Projekte darstellen. Die auf diese Weise zu schaffenden Werte werden aber nicht *nur* ökonomisch-kommerziell bemessen und quantifiziert, sondern letztlich auf dem Boden des Wissens um die Werte der Zweiten Aufklärung; das Wissen um das „Wozu?“ beschränkt sich nicht lediglich positivistisch auf den in Währungseinheiten bilanzierbaren Unternehmenserfolg.

Noch einmal das Fazit: Informatik-Projekte und Wirtschaftinformatik-Projekte werden sicherlich auf eine ökonomische, ressourcensparsame und Ressourceneffizienz fördernde Weise Prozesse und Vorgänge unserer Lebenswelt unterstützen – und eine sehr wichtige, wertvolle und schonungswürdige Ressource ist sicherlich auch die menschliche Arbeitskraft. Wenn aber der Blick so weit in die Ferne gerichtet wird, dass die Möglichkeit einer maschinellen Deckung eines bedeutenden Teils aller Arbeitsnachfrage aufscheint, dann sollte erkennbar werden, dass unter derartigen technischen Gegebenheiten auch einige organisatorische Anpassungen – strukturelle Anpassungen der „sozialen Organisation der Mittelbeschaffung“ – notwendig werden könnten, und es wäre vielleicht an der Zeit, mit entsprechenden Überlegungen zu beginnen.

Literatur

Denning, P.J.; Comer D. E.; Gries, D.E.; Mulder, M.C.; Tucker, A.; Turner, A.J.; Young, P.R. (1992): Computing as a discipline. Comm of the ACM 32, 9-23 (1989); in: *Coy, W.* et al. (Hrsg.): Sichtweisen der Informatik. Braunschweig Wiesbaden 1992

Eversmann, Ludger (2003a): Wirtschaftsinformatik der ‚langen Frist’. Wiesbaden 2003

Eversmann, Ludger (2003b): Die Zukunft der Arbeit als Gestaltungsaufgabe der Wirtschaftsinformatik. Zeitschrift „ARBEIT“ Heft 3/ 2003 (Erscheinungstermin August/September 2003)

Janich, P. (1993): Zur Konstitution der Informatik als Wissenschaft. In: Schefe/Hastedt/Dittrich/Keil (Hrsg.): Informatik und Philosophie. Mannheim Leipzig Wien 1993

Janich, P. (1998): Moderner Konstruktivismus: methodisch-kulturalistisch. Vortrag an der Universität GH Essen anlässlich der Tagung „Wirtschaftsinformatik und Wissenschaftstheorie“ Oktober 1998

Jungclaussen, H. (2002): Kausale Informatik. Wiesbaden 2002

König, W.; Heinzl, A. (2002): Die Wirtschaftsinformatik als Eckwissenschaft der Informationsgesellschaft. In: WIRTSCHAFTSINFORMATIK 44 (2002) 5, S. 508-511

Lorenzen, P. (1991): Philosophische Fundierungsprobleme einer Wirtschafts- und Unternehmensethik. In: H. Steinmann / A. Löhr (Hrsg.): Unternehmensethik. Stuttgart 1991, S. 35 – 67

Lorenzen, P.: (2002): Lehrbuch der konstruktiven Wissenschaftstheorie. Stuttgart; Weimar: Metzler 2000

Mertens, P. (1995): Von den Moden zum Trend. In: König, W. (Hrsg.): Wirtschaftsinformatik 95. Heidelberg 1995, S. 25 ff.

Mertens, P. (2002): Diskussionsbeitrag „Wie viel Wissenschaft(lichkeit) verträgt die Praxis?“ In: WIRTSCHAFTSINFORMATIK 45 (2003) 1, S. 98/99

Müller-Merbach, H. (2002): Die Brückenaufgabe der Wirtschaftsinformatik. In: WIRTSCHAFTSINFORMATIK 44 (2002) 3, S. 300-306

Ortner, E. (1998): Konsequenzen einer konstruktivistischen Grundposition für die Forschung in der Wirtschaftsinformatik. Vortrag an der Universität GH Essen anlässlich der Tagung „Wirtschaftsinformatik und Wissenschaftstheorie“ Oktober 1998

Ortner, E. (2002a): Sprachingenieurwesen – Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik, in: Informatik Spektrum 25 (2002) 1, S. 39-51

Ortner, E. (2002b): Die Zukunft der (Wirtschafts-)Informatik, in: Informatik Spektrum 25 (2002) 5, S. 385-389

Piller, F. T. (2000): Mass Customization. 2. Auflage Wiesbaden 2002.

Schefe, P. (2002): Konstruktivismus nicht konstruktiv – eine Antwort auf E. Ortners Versuch zur „Rekonstruktion“ der Informatik, in: Informatik Spektrum, 25 (2002) 3, S. 230-233

Sen, A. (2002): Ökonomie für den Menschen. Wege zu Gerechtigkeit und Solidarität. Deutsche Ausgabe München 2002

Stiglitz, J. (2002): Die Schatten der Globalisierung. Berlin 2002

Turing, A. (1987): Intelligence Service. Schriften. Berlin 1987

Willke, G. (2002): John Maynard Keynes. Frankfurt 2002

Walthelm, A. (2003): Humans and Robots – A Comparison with Consequences. Künstliche Intelligenz Heft 2 2003, 36-37

Software wird im Handeln zum Hybridobjekt

Rahmen für eine Theorie der Informatik

Dirk Siefkes, Technische Universität Berlin, Informatik

Zusammenfassung: Aus klassischer Informatiksicht ist Informationstechnik ein Dreischritt aus Formalisierung, Algorithmisierung und Maschinisierung. Das ist zu wenig: IT wird in den unterschiedlichsten Arbeitsgruppen, Unternehmen, Institutionen und Gesellschaften entwickelt und eingesetzt. Diese "Kulturen" entstehen und bewegen sich in Prozessen von Schematisierung, Semiotisierung und Organisierung. Bei Entwicklung und Einsatz von IT werden Routinevorgänge in Organisationen durch maschinelle Abläufe ersetzt, im formalen Objekt Software werden dabei menschliche Aktivitäten und maschinelle Abläufe "hybridisiert". Die Ersetzung verändert den Einsatzbereich tiefgreifend und macht Kommunikation und Interaktion in und zwischen solchen Kulturen nötig. Die Arbeit in Informatik verlangt daher Kommunikation und Interaktion mit allen Disziplinen, die sich mit solchen Kulturen und dieser Art Umgang befassen. Aufgabe einer Theorie der Informatik ist, die Wechselwirkungen zwischen den Schritten und die Zusammenhänge zwischen Fachgebieten der Informatik und zwischen Informatik und anderen Disziplinen zu klären.

Informatik und Informationstechnik

Informatik ist eine wissenschaftliche Disziplin; ihr Gegenstand wird heute allgemein Informationstechnik (IT) genannt. IT wird zumeist kommerziell in Organisationen, aber auch in wissenschaftlichen Institutionen entwickelt; die Entwickler sind nicht notwendig als Informatiker ausgebildet. Verwendet wird IT überall; die Kunden und Anwender sind in der Regel keine Informatiker. Wissenschaft, Entwicklung und Verwendung von IT hängen eng zusammen, sind aber drei Bereiche mit unterschiedlichen Interessen und Herangehensweisen.

In den klassischen Lehrbüchern und Curricula erscheint der Gegenstand der Informatik oft als ein Dreisprung von *Formalisierung, Algorithmisierung und Maschinisierung*: Vorgänge werden durch Gesetzmäßigkeiten beschrieben, die Darstellungen in umsetzbare Anweisungen umgeformt und auf die Maschine gebracht. Beim *Formalisieren* bringen wir Aussagen und Vorgänge in eine Form, die unabhängig von Interpretationen, Voraussetzungen, Ansichten ist. Formalisiertes soll allgemeingültig, unabhängig von individuellen Besonderheiten sein. Beim *Algorithmisieren* ersetzen wir Situationen durch Funktionen. Algorithmen sollen unabhängig von der Situation und den Ausführenden eindeutige Ergebnisse bringen. Beim *Maschinisieren* verändern wir Algorithmen so, dass sie von Computern ausgeführt werden können. Maschinen funktionieren unabhängig von lokalen Gegebenheiten. In anderen Selbstdarstellungen werden die drei Schritte zu *Modellierung und Implementierung* zusammengefasst.

Unzureichend sind beide Beschreibungen, weil sie nur die Entwicklung von IT betreffen, die Verwendung zu kurz kommt. Ob die Entwicklung sinnvoll war, zeigt sich erst bei der Verwendung; also erhalten Theorien und Methoden der Entwicklung auch nur dadurch ihre volle Bedeutung.

Informationstechnik und Informatik im Zusammenhang

Um IT sinnvoll zu entwickeln, braucht es also mehr als den Dreisprung; das ist allen Beteiligten klar. Was genau ist aber dieses "mehr"? Und wie ist es wissenschaftlich in den Blick zu nehmen? Das kann nicht in speziellen Theorien der Informatik oder anderer Disziplinen, sondern nur in einer allgemeinen Theorie der Informatik geklärt werden (Coy92, TdI01-03, Sie03).

Als *erstes*: Der Dreisprung geschieht auf dem Boden der entsprechenden Kulturen, setzt Prozesse der *Schematisierung, Semiotisierung und Organisierung* voraus und löst neue aus. Solche Prozesse sind Grundlage von Leben und Kultur: Handlungen laufen in Routinen ab und erzeugen neue Routinen (*Schematisierung*; Pia77, Ste95; Siefkes in Coy92, Bau01, TdI01; Sie02). Erst dadurch werden Handlungen wahrnehmbar und mitteilbar (*Semiotisierung*; Nake in Coy92, Bau01, TdI01), können damit verhandelt und vorgeschrieben werden (*Organisierung*; Gid84, Ort03). Maschinell ausführen können wir nur menschliche Tätigkeiten, die auf diese Weise routinisiert, kommuniziert und organisiert worden sind. Software gestalten heißt, Teile so entstandener Organisationen durch Computersysteme ersetzen (Rol98; Rolf in Coy92, TdI01-03).

Organisationen funktionieren aber anders als Computer: Computer führen programmierte Regeln aus; Menschen nutzen Regeln nur als Rahmen, um sinnvoll arbeiten zu können. Organisationen funktionieren — nicht obwohl, sondern weil die Beteiligten die Regeln regelmäßig verletzen (Ort03, Brö97, Crutzen/Hein in Bau01). Als *zweites* also: Informatiker müssen bei ihrer Arbeit diese Ausgangslage beachten, nicht bloß Beschreibungen in Software gießen. Die Schritte des Dreisprungs werden sinnvoll erst durch ihre Beziehung zueinander und zu Anfang und Ziel. Beim Formalisieren müssen Informatiker von Situationen und deren Beschreibungen abstrahieren, aber nicht absehen (Sie92). Beim Algorithmisieren müssen sie Spielräume fürs Handeln durch Anweisungen fürs Vorgehen ersetzen, ohne die Breite der Möglichkeiten einzuengen. Beim Maschinisieren müssen sie die Systeme zum Laufen bringen, ohne den Boden zu zerstören, auf dem sie laufen. Dafür gibt es in der Softwareentwicklung unterschiedliche Ansätze — ich nenne Partizipation, Objektorientierung, freie Software als Beispiele. Sie werden aber zu selten oder nicht in dem Sinn genutzt, weil sie mühsam und kostspielig sind. Also nicht dem heute obersten Gebot unserer Kultur entsprechen: Effizienz.

Drittens: Da Computersysteme anders funktionieren als menschliche Organisationen, verändert sich beim Einsatz von IT nicht nur der ersetzte Teil. Man kann ihn nicht herausschneiden und spurenlos durch eine Maschine ersetzen; die ganze Organisation wird in Mitleidenschaft gezogen. Auch wenn die Regeln nicht geändert wurden, müssen alle, die mit dem maschinellen System in Berührung kommen, neue Verhaltensformen aufbauen. Das ist ein langwieriger, oft schmerzhafter Prozeß — nicht nur für die Herausgeschnittenen —, der aber in der Informatik zu wenig Aufmerksamkeit findet (Brö97, Brödner et al. in TdI02, 03, Hrachovec in Bau01, Wilkens in TdI02, Sesink in TdI03). In Softwaretechnik und Theoretischer Informatik werden Theorien und Methoden vor allem für die Entwicklung, nicht für die Verwendung von IT-Systemen produziert. Verwendung ist Sache der Angewandten Informatik; dort fehlt i.a. aber der Blick auf die Entwicklung (Rolf in TdI01-03).

Als *viertes* müssen Informatiker daher die Augen für die kulturellen Böden öffnen, auf denen sie agieren und die sie damit stärken oder schwächen (Siefkes et al. in TdI02). Als Wissenschaftler müssen sie sich fragen, welche Entwicklungen sie fördern wollen. Als Anwender müssen sie mit den unterschiedlichen Kulturen kommunizieren und interagieren, in denen sie Computersysteme einsetzen wollen. Sie werden nur soweit fruchtbare Veränderungen erreichen, wie sie ihre Mikrosicht mit der Makrosicht auf die Umgebungen abstimmen (Rol98).

IT-Einsatz als Entwicklung

Der Transformation von Beschreibungen, die von Menschen zu interpretieren sind, in Befehle, die vom Computer ausgeführt werden, dient der eingangs erwähnte Dreisprung der IT. Er *vermittelt* also nur zwischen den beiden Bewegungen "vor Ort": den Veränderungen der Situation vor und nach Einführung des IT-Systems (Sesink in TdI03). Die beiden Schritte werden verharmlosend *De- und Rekontextualisierung* genannt. Eine soziale Situation ist aber kein Text, den man mit *cut and paste* spurenlos verändern könnte. Wie im letzten Absatz beschrieben werden beim De- und Rekontextualisieren auf vielfältige Weise Schemata, Bezeichnungen und Regelungen eingeführt, verändert oder beseitigt; die bestehende soziale Situation wird erst zerstört und richtet sich dann neu ein. Diese beiden Schritte sind das A und O jeden IT-Einsatzes, hier fügt sich der Dreisprung zum Zyklus bzw. zur Spirale. Der Boden, auf dem der Zyklus läuft, entwickelt sich dabei durch die kulturellen "Schritte". IT-Systeme werden nicht erst entwickelt und dann verwendet; zu ihrer vollen Bedeutung entwickeln sie sich erst durch die Nutzer. Software gestalten heißt beides einbeziehen (Grü00, Rolf in TdI01-03).

In der Praxis laufen die Schritte nicht nacheinander ab, sondern parallel gegeneinander versetzt oder bunt durcheinander; das ist in der Softwaretechnik ein viel behandeltes Thema (Flo92, Fischer in TdI03). Das liegt nicht nur daran, dass die Wirklichkeit schmutzig und unordentlich ist. Die Schritte hängen eng zusammen, beeinflussen, verstärken, behindern sich, müssen also aufeinander Rücksicht nehmen, müssen rückgängig gemacht und neu durchgeführt werden. Insbesondere beziehen sich alle Schritte auf den ersten und letzten; also muß sich auch ihre wissenschaftliche Grundlegung daran ausrichten.

Eine Theorie der Informatik muß sich also mit allen Schritten des Zyklus, vor allem aber mit den Zusammenhängen zwischen ihnen und den Schritten "am Boden" bei der Entwicklung und Verwendung von IT befassen. In einem Interdisziplinären Forschungsprojekt "Sozialgeschichte der Informatik" haben wir Informatik aus dieser Sicht historisch charakterisiert: Für Informatiker beschreiben Computerprogramme gleichzeitig die Vorgänge, die sie auf den Rechner bringen wollen, und die Operationen, die der Rechner ausführen wird; Frieder Nake nennt solche Texte *algorithmische Zeichen*, Computer daher *semiotische Maschinen*, die Informatik *technische Semiotik* (Nake in Coy92, Bau01 und in diesem Band). Da wir Beschreibung und Beschriebenes gewohnheitsmäßig identifizieren, scheinen Programme *selbstagierend*: scheinbar eigenständig führen sie menschliche und maschinelle Aktionen gleichzeitig aus; Michaela Reisin und Christiane Floyd nennen Software daher *autooperationale Form* (Rei92, Flo97). Auf diese Weise *hybridisieren* Informatiker Mensch und Computer, organisiertes Handeln und maschinelle Abläufe, mit Hilfe formaler Notationen und Modelle. Software wird im Handeln der Subjekte zum *Hybridobjekt* (SGI98, Eul99, Sta01, Sie02, Siefkes in Bau01, TdI02, 03). Eine Theorie der Informatik muß Gestaltung aus dieser Sicht verstehen.

Teile einer Theorie der Informatik

Eine Theorie der Informatik kann also keine philosophische Theorie sein. Es genügt nicht, die Informatik als Gebiet festzulegen, ihre Inhalte und Methoden zu bestimmen, ihre Grenzen abzustecken. Solche Definitionen der Informatik können das Ergebnis unserer Arbeit sein; wenn wir damit beginnen, verbauen wir das Gelände, das wir übersichtlich und begehbar machen wollen. Die Arbeit an einer Theorie der Informatik kann nur aus der informatischen Arbeit heraus entstehen, wenn sie ihr eine Hilfe sein soll (Coy92, TdI01-03).

Als Beispiel kann die *Theoretische Informatik* dienen. Sie liefert mathematische Theorien und Modelle fürs Spezifizieren und Berechnen und für den Computer, also für den Dreisprung, und könnte daher mathematischer Teil einer allgemeinen Theorie der Informatik sein. Sie unterscheidet sich von entsprechenden Gebieten der Mathematik, soweit sie die Schritte als eingebettet in den Zyklus "Entwicklung und Verwendung von IT", also als Teil eines Hybridisierungsvorgangs sieht. Solche Untersuchungen könnten die Auswahl von Forschungsgegenständen und -methoden und damit die Forschungsrichtung der Theoretischen Informatik beeinflussen. Das würde ihr ein besseres Heimatrecht in der Informatik verschaffen, ohne dass sie ihre mathematische Identität leugnen müßte, und würde sie in eine allgemeine Theorie der Informatik einbetten. Dasselbe trifft in schärferer Form für die *Technische Informatik* zu, die sich gern als technisch orientiert von der Informatik absetzt und durch einen theoretischen Bezug besser integriert wäre.

Ähnlich könnte es für die anderen Schritte aussehen, mit unterschiedlichen Bezügen zu den beteiligten Disziplinen. *Softwaretechnik* ist ein, wenn nicht *das* zentrale Gebiet der Informatik. Zu einer Theorie der Informatik könnte sie beitragen, wenn sie den hybriden Charakter von Software und damit Fragestellungen einbezieht wie (Flo92, Brödner et al. in TdI02, 03, Rolf in TdI01-03): Nach welchen (nicht nur technischen) Maßstäben bewerten wir die Qualität von Software? Welche anderen können wir wählen? Welche Arten von Arbeitsvorgängen können durch welche Formen von Software(gestaltung) unterstützt oder ersetzt werden, welche werden eher behindert? Was ist der Grund für Softwarehavarien? Wie hängt die Gestaltung von Software mit der Gestaltung von Organisationen durch IT-Einsatz zusammen? Für welche Aspekte unserer Arbeit als Softwaretechniker können wir die Verantwortung übernehmen?

Ebenso wichtig für eine Theorie der Informatik sind andere Gebiete, die eine wichtige Rolle bei Entwicklung und Einsatz von IT spielen, aber aus historischen Gründen in der Informatik keinen Fuß gefaßt haben. Eine Soziologie, Psychologie, Semiotik, Linguistik, Pädagogik, Philosophie oder Geschichte der Informatik wird es als Fachgebiet der Informatik kaum geben, obwohl es teilweise intensive Bemühungen um sie gibt; zur Geschichte s.o. und z.B. (Hel03). Analysen der Informatik innerhalb dieser Disziplinen können aber nur zu entsprechenden Theorien *über* die Informatik führen, die für die Informatik ziemlich folgenlos blieben. Denn für Maschinen und Formalismen gilt in besonderem Maß, was allgemein fürs Lernen anerkannt ist: Verstehen erwächst nicht allein aus dem Umgang mit der Sache oder der Reflexion über sie, sondern nur aus der Wechselwirkung beider. Entsprechende Theorien *der* Informatik werden daher am ersten von Informatikern zusammen mit Vertretern der anderen Disziplinen erarbeitet werden. Nur dann können sie als Teile einer allgemeinen Theorie der Informatik unsere Disziplin befruchten. Als Beispiele mögen die erwähnten Arbeiten von Frieder Nake zur Informatik als Technischer Semiotik, von Arno Rolf, Peter Brödner und Gerhard Wohland zur Gestaltung von IT-Systemen in Organisationen und Gesellschaft, von Werner Sesink und Ulrike Wilkens zur Medienpädagogik, von Peter Eulenhöfer u.a. zur Sozialgeschichte der Informatik, von mir zu einer evolutionären Theorie geistiger und sozialer Entwicklung sowie weitere interdisziplinäre Arbeiten in TdI01-03 und Bau01 dienen.

Rahmen für eine Theorie der Informatik

Eine Theorie der Informatik ist aber mehr als die Sammlung dieser und anderer Teiltheorien. So wie die sechs Schritte im Zyklus der Entwicklung und Verwendung von IT ihre Bedeutung erst als Teile im Zusammenhang und durch ihre wechselseitigen Abhängigkeiten erhalten, erhalten die zugehörigen Teiltheorien ihre Bedeutung erst durch die Beziehungen zueinander.

Aufgabe einer Theorie der Informatik ist also, die widersprüchlichen Beziehungen zwischen diesen unterschiedlichen Fachkulturen herauszuarbeiten; für eine Reihe von Fächern habe ich das Unterfangen in (Sie03) begonnen. Die Beziehungen können als Kooperation oder Konflikt, als Unterstützung oder Konkurrenz zu Tage treten und müssen als solche aufgenommen werden. Es geht nicht darum, das Verbindende in den unterschiedlichen Gebieten zu finden, sie auf ein gemeinsames Ziel festzulegen oder gar anzugleichen. Die Kreativität einer Gruppe speist sich aus ihrer Vielfalt; dazu muß sie sich als Gruppe wahrnehmen. Eine Theorie der Informatik hat also insbesondere die Aufgabe, die Identitätsfindung von Informatikern zu analysieren und zu stärken, Selbst- und Fremdeinschätzung zu vergleichen und mit Arbeitsweisen und Ausbildungszielen zu korrelieren (Bath et al. in TdI03, Siefkes et al. in TdI02).

Literatur

(Bau01) K. Bauknecht et al. (Hg.): Informatik 2001, Jahrestagung GI & OCG. Workshop "Erkenntnistheorie - Semiotik - Ontologie". Österreich. Computergesellschaft, Wien.

(Brö97) Peter Brödner: Der überlistete Odysseus. Über das zerrüttete Verhältnis von Menschen und Maschinen. Berlin: edition sigma.

(Coy92) Wolfgang Coy et al. (Hg.): Sichtweisen der Informatik. Vieweg.

(Eul99) Eulenhöfer, Peter 1999: Die formale Orientierung der Informatik. Zur mathematischen Tradition der Disziplin in der Bundesrepublik Deutschland. Dissertation, FB Informatik, TU Berlin.

(Flo92) Christiane Floyd et al. (ed.): Software Development and Reality Construction. Springer.

(Flo97) Christiane Floyd: Autooperationale Form und situiertes Handeln. In C. Hubig (Hrsg.): Cognitio humana — Dynamik des Wissens und der Werte. XVII. Deutscher Kongreß für Philosophie, Vorträge und Kolloquien, S. 237-252.

(Grü00) Barbara Grüter: e-motion - über elektronische Formen der Bewegung und die Gestaltung von Interaktionssystemen. MMI-Interaktiv, Nr.4, S. 1-16, ISSN 1439-7854. http://www.mmi-interaktiv.de/ausgaben/

(Gid84) Anthony Giddens: The Constitution of Society. Outline of the Theory of Structuration. Berkeley. — Deutsch: Die Konstitution der Gesellschaft. Campus 1988.

(Hel03) Hans Dieter Hellige: Geschichten der Informatik. Erscheint bei Springer.

(Ort03) Günther Ortmann: Regel und Ausnahme. Paradoxien sozialer Ordnung. Suhrkamp.

(Pia77) Jean Piaget: The Essential Piaget. H.W. Gruber & J. Vonèche (eds.), Basic Books.

(Rei92) Fanny-Michaela Reisin, Kooperative Gestaltung in partizipativen Softwareprojekten. Peter Lang, Frankfurt/Main.

(Rol98) Arno Rolf: Grundlagen der Organisations- und Wirtschaftsinformatik. Springer.

(SGI98) Dirk Siefkes et al. (Hg.): Sozialgeschichte der Informatik. Kulturelle Praktiken und Orientierungen. Deutscher Universitätsverlag.

(Sie92) Dirk Siefkes: Formale Methoden und kleine Systeme. Lernen, leben und arbeiten in formalen Umgebungen. Vieweg.

(Sie02a) -"- : Sozialgeschichte und kulturelle Theorie der Informatik. TU Berlin, Fak. Elektrotechnik & Informatik, Bericht 02-16.

(Sie02b) -"- : Hybridization in Computer Science. TU Berlin, Fak. Elektrotechnik & Informatik, Bericht 02-17.

(Sie03) -"- : Rahmen für eine Theorie der Informatik. Manuskript.

(Sta01) Heike Stach: Zwischen Organismus und Notation. Zur kulturellen Konstruktion des Computer-Programms. Deutscher Universitätsverlag.

(Ste95) Daniel N. Stern: The Motherhood Constellation. Basic Books. Deutsch: Die Mutterschaftskonstellation. Klett-Cotta 1998.

(TdI01) Frieder Nake, Arno Rolf, Dirk Siefkes (Hg.): Informatik - Aufregung zu einer Disziplin. Tagung Theorie der Informatik 2001. Uni Hamburg, FB Informatik, Bericht 235. http://tal.cs.tu-berlin.de/siefkes/Heppenheim

(TdI02) -"- : Wozu Informatik? Theorie zwischen Ideologie, Utopie, Phantasie. Tagung Theorie der Informatik 2002. TU Berlin, Fak. Elektrotechnik & Informatik, Bericht 02-25. http://tal.cs.tu-berlin.de/siefkes/Hersfeld

(TdI03) -"- : Informatik zwischen Konstruktion und Verwertung. Tagung zur Theorie der Informatik 2003. Uni Bremen, FB Mathematik & Informatik, Bericht in Vorbereitung.

Unser aller Profession gib uns heute …
oder die Frage nach einer mäeutischen Informatik

Peter Bittner

Institut für Informatik
Humboldt-Universität zu Berlin
Unter den Linden 6
10099 Berlin
bittner@informatik.hu-berlin.de

Abstract: In der bisherigen Professionalisierungsdebatte der Informatik fällt ein deutlicher Bezug zu einem klassisch-kriteriellen bzw. funktionalistischen Professionsbegriff auf, der sich seit den 30er Jahren (nicht nur) in der Professionssoziologie als sehr wirkmächtig darstellt. Ein solcher Professionsbegriff verstellt aber den Blick auf die Problemlagen, die von Interesse sind, wenn wir professionelles informatisches Handeln (im Sinne angewandter Informatik) begreifen wollen. Dieser Beitrag will den Blick öffnen – hin zu anderen Professionsbegriffen und anderen Professionalisierungsdiskursen, die den Denk- und Handlungsmustern der angewandten Informatik näher liegen.

1. Vorbemerkungen

Diese kleine Arbeit[1] steht in einer Serie von Arbeiten (Bittner 2002a, 2003a, 2003b), die zusammen drei übergeordnete Konstruktionsprinzipien von Theorien (angewandter) Informatik markieren sollen und dem Theorie-Diskurs „querdenkende" Impulse geben möchten. Ich möchte diese theoretischen Überlegungen weitgehend „pragmatisch" orientieren. Im Mittelpunkt dieser übergeordneten Untersuchung steht die Frage der Vermittlung der Informatik in ihrer lebensweltlichen Praxis. Wie kann dies konzeptionell (Allgemeine Informatik), praktisch (Handlungsorientierung) und bezogen auf das professionelle Handeln (Mäeutische Informatik) geschehen?

Hier möchte ich der Frage nachgehen, wie man einen adäquaten Begriff von Profession finden könnte, um die Frage der Professionalisierung der angewandten Informatik angemessen zu behandeln. Es soll der Blick[2] auf Ansätze[3] der Professionssoziologie und aus der Professionalisierungsdebatte in der Pädagogik gelenkt werden.

[1] Sie basiert auf einem Vortrag *Unser aller Profession gib uns heute ...*, den ich im Rahmen der Sitzung des Arbeitskreises „Verantwortung und Informatik" des FB IUG „Informatik und Gesellschaft" der Gesellschaft für Informatik am 15.03.2002 an der Humboldt-Universität zu Berlin gehalten habe. Unter gleichem Titel ist dann eine erste Textfassung dieses Beitrages als „Positionspapier" erschienen als (Bittner, 2002b). Aus Anlass des Symposiums wurde der Beitrag z.T. neu gefasst, ergänzt und überarbeitet.

2. Bisheriger Bezugspunkt: Klassisch-kriterieller Professionsbegriff

Die bisherigen Äußerungen in der Professionalisierungsdebatte der Informatik beziehen sich auf einen klassisch-kriteriellen Professionsbegriff. Dies gilt sowohl für die angloamerikanischen Arbeiten (z.B. Denning, 2001; Ford & Gibbs, 1996), wie auch für Arbeiten aus dem deutschsprachigen Raum (z.B. Schinzel & Kleinn, 2001; Hartmann, 1995).

2.1 Ein kurzer professionssoziologischer Exkurs

Bestimmend in der Professionssoziologie sind klassisch-kriterielle (und funktionalistische) Ansätze, wie man sie z.B. bei Parsons (1939) oder Goode (1957, 1972) findet. Professionen sind dabei im wesentlichen akademische Berufe (mit langer spezialisierter Ausbildung), bei denen eine Steigerung von Rationalität bei der Verwirklichung von Handlungszielen feststellbar ist. Sie sind markiert durch eine deutliche Begrenzung der Kompetenz, die durch die Aufgabenstellung und das Problem des Klienten (Ausrichtung auf wichtige individuelle oder kollektive Probleme) definiert ist. Professionelles Handeln sei nicht von partikularen Interessen (z.B. Sympathie oder Antipathie) geprägt. Die hierfür notwendige hohe Autonomie der Professionellen schlägt sich nach Goode nieder:

- im Recht, den eigenen Nachwuchs zu bilden und zu erziehen,
- im Recht der professionellen Selbstkontrolle und
- bei der (autonomen) Strukturierung des professionellen Berufsalltags.

Hinzu kommt eine spezielle Ethik, die den Schutz der Klienten in ihrer jeweiligen Situation vor Ausbeutung sichert und eine Selbstverpflichtung der Professionsmitglieder beinhaltet.

[2] Hierzu bedarf es einiger methodischer Überlegungen, darum sei die dem Beitrag zugrundeliegende – etwas ungewöhnliche – Verfahrensweise eingehender erläutert. Bezogen auf die Fragestellung verlasse ich zunächst den Rahmen in der Informatik bekannter Methoden/Denkweisen, um mir das angrenzende Fachgebiet der Professionssoziologie zu erschließen. Die notwendige Aneignung wird abgesichert durch ein intensives Literaturstudium und (auch persönlichen) wissenschaftlichen Austausch mit Vertretern und Vertreterinnen der entsprechenden *Scientific Community*. In der Folge diskutiere ich meine Übertragungen und aufgefundenen „ähnlichen" Befunde mit Blick auf die Informatik in der anderen Community, um dann die (vorläufigen) Ergebnisse wieder in passende Informatik-Diskurse rückzuvermitteln. An dieser Stelle kann ich nun den Versuch unternehmen, in der eigenen Disziplin zu überprüfen, ob die auf anderem Gebiete gewonnenen Erkenntnisse, im Rahmen der (andauernden) Selbstverständigungsdebatte zu Widersprüchen führen oder nicht. Das Ausbleiben von Einwänden deute ich als Bestätigung der vorgebrachten Thesen bzw. gefundenen Analogien. Unabhängig davon ist natürlich zu untersuchen, inwieweit die Ergebnisse aus Sicht der Praxis (bzw. angewandten Informatik) belastbar sind und welche Konsequenzen für die Praxis hieraus zu ziehen sind – für die unternommene Untersuchung ist dieser Teil aber noch *work-in-progress*.

[3] ... Profession zu verstehen

2.2 Kritik aus der Informatik

Schinzel & Kleinn (2001) u.a. arbeiteten sich an der Nicht-Erfüllung von (klassischen) Professionskriterien ab. Schon die wenigen nachfolgend aufgeführten Argumente deuten an, dass die Informatik (bezogen auf diese Kriterien) als nicht professionell angesehen werden muss.

- Zum für Professionen vielfach eingeforderte „core of the discipline“: Die Informatik hat noch keinen stabilen Kern (informatischer Grundlagen) entwickelt. Inwieweit dies – abgesehen von der Theoretischen Informatik – (überhaupt) möglich sein könnte, ist (zumindest bisher) nicht geklärt.
- Für Professionen wird ein „abgegrenztes“ Arbeitsfeld gefordert: Gegenwärtig erschließt sich die Informatik beständig neue Tätigkeitsfelder. Andersherum bewegen sich verschiedene Anwendungsgebiete auf die Informatik zu.
- Für Professionen wird hohes Maß an Autonomie gefordert: In vielen Projekten unterliegen „IT-Professionals“ jedoch einem extremen zeitlichen Druck. Dies führt zu unseriösen Analysen, dem Kunden zu früh übereigneten Produkten, ungenügender Berücksichtigung selbst gesetzlich vorgeschriebener Pflichten (u.a. Hornecker & Bittner, 2000; Ford & Gibbs, 1996).
- Professionen gründen sich auf einer „langen“ akademischen Ausbildung: Ein Zugang zu informatischen Berufen ist heute ohne Weiteres auch ohne eine Hochschulausbildung, z.T. gar ohne (formale) Ausbildung möglich. Ein „Wissensmonopol“ ist nicht vorhanden. Ob diese Schließung gewünscht ist, wäre zu diskutieren. Quereinsteiger können auch bedeutsam sein, wenn sie den notwendigen Anwendungsbezug in Projekte einbringen.

2.3 Kritik aus der Professionssoziologie

Jahrzehntelang stand die Professionssoziologie fest und ziemlich unerschüttert in der Tradition von Carr-Saunders und Wilson (1933), die Waddington (1996) „checklist approach“ oder auch „trait-approach“ nannte. Also den Professionsbegriffen, die wir zuvor als klassisch-kriteriell bzw. funktionalistisch kennenlernten.

Bei unseren weiteren Debatten zur Professionalisierung der Informatik sollten wir nicht (vor-)schnell Bezüge zu diesem klassisch-kriteriellen (oder funktionalistischen) Professionsbegriff herstellen. Hierzu liefert uns die professionssoziologische Forschung verschiedene Argumente, u.a.:

- Während sich in den USA Berufe eher bottom-up professionalisieren, geschieht dies im deutschsprachigen Raum eher top-down (vgl. Koring, 1999:Teil 6.4). Aufgrund dieser strukturellen Differenzen ist eine einfache Übernahme anglo-amerikanischer Professionsbegriffe nicht ohne Weiteres möglich.
- Durch die erhöhte Selbstkontrolle und kollegiale Kontrolle sind Professionen unempfindlicher für Laienkritik und eine „gesellschaftliche Kontrolle“. Eine Schließung, die wir im Sinne einer *Allgemeinen Informatik* vermeiden sollten (vgl. Bittner, 2003a; Bittner, 2003b)[4].
- Anhand von Kriterienlisten lassen sich komplexe Identitäten von Gruppen, die mit einer Vielzahl von Adressaten sowie ihren Trägern und der ganzen Gesellschaft interagieren, kaum (er-)fassen.
- Im Rahmen klassisch-kriterieller Professionsbegriffe wird über die Tätigkeit der Professionellen und die *zugehörigen Denk- und Handlungsmuster* nur wenig ausgesagt.

3. Auseinandersetzung mit anderen Professionsbegriffen

Die Auseinandersetzung kann im Rahmen dieser Arbeit (aus Platzgründen) nur exemplarisch geführt werden. Sicherlich ließen sich aus den verschiedenen Varianten von Professionsbegriffen[5] bzw. aus den verschiedenen Professionalisierungsdebatten (Medienpädagogik, Sozialberufe), die aus der Literatur bekannt sind, Erkenntnisse für unsere Frage gewinnen. Zunächst sollen hier ein strukturtheoretischer Professionsbegriff (Oevermann) und die Professionalisierungsdebatte der Pädagogik (hier v.a die Arbeiten von Koring) Gegenstand der Untersuchungen sein.

3.1 Ein strukturtheoretischer Professionsbegriff – gedeutet für die Informatik

Die Arbeiten Oevermanns (1978, 1983, 1996) markieren eine *professionssoziologische Wende*[6]. Den Professionen werden innerhalb dieses Ansatzes drei zentrale gesellschaftliche Aufgaben zugewiesen:

- Sie seien zum einen mit der kritischen Prüfung von Wahrheitsbehauptungen zu beschäftigen, wie dies in der Wissenschaft der Fall ist.
- Sie seien für die Beschaffung von Konsens und Konformität zuständig, wie dies bei Richtern und Rechtsanwälten, teilweise auch bei Politikern der Fall ist.
- Sie hätten für die Bereitstellung therapeutischer Leistungen (um Menschen gesund, handlungsfähig und orientierungsfähig zu halten) zu sorgen, wozu Ärzte, aber auch Priester, Lehrer und Sozialpädagogen zählen.

---

[4] *Allgemeine Informatik* sei charakterisiert durch (1) die Einstellung, Informatik für die Allgemeinheit zu öffnen, sie prinzipiell lernbar und kritisierbar zu machen, (2) die Darstellung informatischer Entwicklungen in ihren Sinngebungen, Bedeutungen und Bedingungen, (3) die Vermittlung der Informatik in ihrem lebensweltlichen Zusammenhang über die Fachgrenzen hinaus, (4) die Auseinandersetzung über Ziele, Verfahren, Wertvorstellungen und Geltungsansprüche der Informatik (in Anlehung an Wille (1996, 1988) und von Hentig (1972)). Ich bitte dies nicht mit einer allgemeinen Informatik zu verwechseln, die ihre Inhalte/Vorgehensweisen als unspezifisch gegenüber den verschiedensten Anwendungsgebieten versteht oder das versucht in den Blick zu nehmen, was jeder Informatiker wissen müsse.

[5] In der Literatur finden sich zumindest Beschreibungen indikatorischer (kriterieller), funktionalistischer, strukturtheoretischer, systemtheoretischer, symbolisch-interaktionistischer und machttheoretischer Sichten auf Profession.

[6] Oervermanns Ansatz wird zumeist als *strukturtheoretisch*, zuweilen auch als *deduktiv* charakterisiert.

Wahrheit, Konsens und Therapie werden dabei bei Oevermann als die wesentlichen Funktionsvoraussetzungen jeder Gesellschaft verstanden. In jeder Profession spielen alle drei Aspekte eine gewisse Rolle. Es gibt aber Spezialisierungen. So wird z.B. die Pädagogik von Oevermann im Bereich der Therapiebeschaffung angesiedelt.

Für Oevermann verbinden sich in der *realisierten Professionalität* die:

- *wissenschaftliche Kompetenz*, die den Umgang mit Theorie und den engen Kontakt zum Fachwissen der Disziplin (Jura, Medizin, Theologie oder Erziehungswissenschaft) betrifft und die
- *hermeneutische Kompetenz*, aufgrund derer ein bestimmtes Problem verstanden werden kann. Dazu ist wissenschaftliches Wissen allein nicht ausreichend; praktische Erfahrung ist notwendig.

Das wissenschaftliche Wissen ist nicht umstandslos auf praktische Probleme anwendbar, weil es abstrakt ist und die Wirklichkeit demgegenüber sehr konkret und vielfältig. Die intuitive Auseinandersetzung ist nicht ausreichend, weil Intuition und Erfahrung sich im Kreise drehen können. Die Aufgabe des Professionellen besteht darin, zum Zweck der Bearbeitung eines Problems, *das wissenschaftliche und das hermeneutisch-fallbezogene Wissen, so zu verbinden, dass praktische Deutungen und Handlungsstrategien zustande kommen.*

Ganz ähnlich die Position Nohls (2002/1933), der versucht, Kriterien für die Angemessenheit pädagogischen Handelns zu entwerfen. Versucht man dessen Argumentation auf die Informatik zu übertragen, dann wird der Informatiker[7] zu einer Vermittlungsinstanz zwischen Subjektivität (Perspektive des Klienten/Adressaten) und Objektivität (gesellschaftliche Anforderungen). Sein Kennzeichen ist, dass er aufgrund wissenschaftlicher und praktischer Kenntnisse zum einen *auswählend* und *vermittelnd*, zum anderen *interpretierend* tätig ist.

Eine zentrale Stellung nimmt im professionellen Handeln die *stellvertretende Deutung* ein. Professionelle deuten für Klienten[8] (Dienstleistungssicht!) ein Problem, das der Klient selbst nicht verstehen und lösen kann, weil er von dem Problem betroffen ist. An dieser Stelle ergeben sich wesentliche Fragen an die Informatik: Wie handeln hier Informatiker? Wie verhält es sich in diesem Sinne mit partizipativen Verfahren? Können bzw. dürfen Informatiker handeln, wenn ein Klient sein Problem nicht (hinreichend) versteht?

Folgte man nicht der objektiven Hermeneutik[9] von Oervermann sondern systemtheoretischen Denkansätzen, so kommt man mit Stichweh (1992) zu ähnlichen Ergebnissen. Dieser nimmt die Beziehung zwischen Professionellen und Klienten aus systemtheoretischer Perspektive in den Blick. Er geht der Frage der *Interaktionsabhängigkeit* in der Beziehung Professionelle – Klienten nach und davon aus, dass Probleme der Personen

[7] Bei der Verwendung des Begriffs Informatiker seien die Informatikerinnen mitbedacht. Letztere mögen die Verwendung der männlichen Form entschuldigen.
[8] Insofern kann dies nur Gültigkeit beanspruchen, wenn der Professionelle dem Klienten auch zugewandt ist!
[9] der diesen Überlegungen zugrunde liegenden philosophischen Denkweise

als Ort der Problembearbeitung „Interaktionssysteme“ benötigen. Seine Kategorie der Vermittlung[10], die an Oevermanns stellvertretende Deutung erinnert, steht im Mittelpunkt der Professionstheorie und der Handlungswirklichkeiten der Professionen. Professionen sind mit *kulturellen Sachthematiken* befaßt, „von denen ihre Klientel strukturell [...] durch eine erhebliche Distanz getrennt ist, und weil die jeweilige Profession [...] immer auch Distanzüberbrückungen intendiert“ (Stichweh, 1992:34).

Professionen sind nach Oevermann, Nohl und Stichweh als wichtige gesellschaftliche Orte der Vermittlung von Theorie und Praxis in der modernen Welt anzusehen. Professionelle haben diese Vermittlung konkret bei jedem bearbeiteten Fall (neu) zu leisten. Mit dieser Art Professionsbegriff sind wir (bestimmten) Handlungsformen angewandter Informatik sehr viel näher, als dies mit einem klassisch-kriteriellen Professionsbegriff jemals möglich gewesen wäre.

3.2 Lernen aus der Professionaliserungsdebatte der Pädagogik: Zur Vorstellung einer *Mäeutischen Informatik*

In Auseinandersetzung mit den Arbeiten Oevermanns ist Korings (vgl. 1999:Teil 6.8) Bild professioneller Pädagogik geprägt von zwei regulativen Ideen, die sich sinngemäß auf die Informatik übertragen lassen:

- Der Informatiker hat sich an der Ermöglichung von Selbsttätigkeit und Selbständigkeit der Klienten zu orientieren.
- Er hat sich an der Struktur einer mäeutischen Informatik zu orientieren, also einer Informatik, die an schon vorhandene Kompetenzen produktiv anknüpft.

Der Informatiker muss also mit situativen Arrangements dafür sorgen, dass Selbsttätigkeit möglich ist und gefördert wird. Der Klient muss sich produktiv mit dem, was entstehen soll (Informatik-System) und den kulturellen Veränderungen befassen – ansonsten ist die *anwaltliche*[11] Aufgabe des Informatikers nicht wahrnehmbar.

Dies führt uns geradewegs zur *Mäeutik* als (ehemals pädagogischer) Hebammenkunst. Für Informatiker in der *Dienstleistungssituation* heißt dies, darin geschult zu sein, im (dialektischen) Gespräch ein Wissen bzw. Können zutage zu fördern, das dem Gegenüber zunächst verborgen war. Im professionellen Handeln strukturiert und begleitet der Informatiker den Prozess, in welchem die Klienten versuchen, die Probleme und Bedingungen ihres eigenen „Arbeitens“ zu artikulieren. Der Informatiker deutet diese artikulierte neue Bedeutung in ihrem Verhältnis zum Thema, zum Problem, zur Person und zum Gestaltungsprozess selbst. An diesen informatischen Deutungen können die Adressaten erkennen, an welcher Stelle sie im Gestaltungsprozess stehen.

[10] Stichweh zieht den Begriff der Vermittlung dem der stellvertretenden Deutung vor, weil stellvertretende Deutung eine zweistellige Relation suggeriere, während tatsächlich eine dreistellige vorliege, und zwar im Sinne der Repräsentation einer autonomen Sinnperspektive oder Sachthematik durch den Professionellen in seinem interaktiven Verhältnis zum Klienten (vgl. Darstellung Stichweh in Koring, 1996:320).
[11] in Bezug auf die stellvertretende Deutung

4. Wie weiter?

Es zeigt sich, dass wir aus dem professionssoziologischem Diskurs und der Professionalisierungsdebatte der Pädagogik lernen und Anregungen für unser Verständnis professionellen informatischen Handelns gewinnen können. Diese Diskussion steht für die Informatik noch am Anfang. Verschiedene andere Professionsbegriffe und weitere Professionalisierungsdiskurse sind noch nicht oder nur wenig darauf hin untersucht, ob sie für die in diesem Beitrag gestellten Fragen nutzbar gemacht werden können.

Der Ausdeutung *stellvertretender Deutung* im informatischen Handeln kommt ein gewisses Gewicht zu. Die These, dass uns ein strukturtheoretischer Professionsbegriff bei der Selbstverständigung über das informatische Handeln helfen kann, lässt sich m.E. kaum von der Hand weisen. So untersucht Hofer (2002) z.B. die „Beratungskomponente in der Softwareentwicklung im Spannungsfeld von technischer Problemlösung und stellvertretender Krisenbewältigung". Mit der Frage, inwieweit Oevermanns Professionalisierungsmodell auf die Informatik übertragbar ist, wird auch die Frage akut, was stellvertretende Deutung im Kontext der Software-Entwicklung bedeutet und wie mit dem Verständnis des Anwenders im Prozess der Formalisierung umgegangen wird.

Mit der Kenntnisnahme o.g. Diskurse gerät auch das Theorie-Praxis-Verhältnis der Informatik wieder in den Blick, also das Verhältnis zwischen der Informatik als wissenschaftlicher Disziplin und der Informatik in der beruflichen Praxis; genauer das Verhältnis zwischen Empirie und Reflexion und damit auch die Frage nach der Hermeneutik informatischer Problemlagen, um so zu einer materialen Definition dessen zu kommen, was informatische Professionalität heißen könnte.

Die Ergebnisse sind dann weiterhin mit den Arbeiten in Beziehung zu setzen, die sich – z.B. von der Anforderungs- oder Systemanalyse her kommend – mit dem Verhältnis Informatiker ↔ Klient auseinandersetzen oder die Arbeitsbedingungen von den IT-Professionals untersuchen, die mit dem „Anderen" der Informatik konfrontiert sind.

Danksagung

Mein Dank für einen anregenden, sehr offenen und intensiven Austausch über professionssoziologische Fragestellungen, Denk- und Handlungsmuster gilt Dr. Michaela Pfadenhauer (St. Gallen) und Andreas Franzmann (Frankfurt), aber auch Prof. Dr. Harald Mieg (ETH Zürich) und Prof. Dr. Dr. Manfred Moldaschl (TU Chemnitz). Viel gelernt über die Professionalisierungsdebatte der Pädagogik habe ich durch die Arbeiten von Prof. Dr. Bernhard Koring (TU Chemnitz). Auf der 3. Arbeitstagung „Theorien der Informatik" (03.-05.04.2003, Bad Hersfeld) wurde in einem der Arbeitskreise intensiv über die Frage professionellen informatischen Handelns diskutiert – den Mitstreitern dort sei ebenfalls gedankt. Den Gutachtern verdanke ich vielfältigste Hinweise, die sicherlich zur Verbesserung der Arbeit beigetragen haben, auch wenn einige Einwände für diese Textfassung unberücksichtigt geblieben sind.

Literaturverzeichnis

[Bi02a] Bittner, Peter: Theorien der Informatik und Kritische Theorie. Über die Vermittlung zweier Denkwelten. In: Gehrlein, Ulrich; Krebs, Heike; Pfeiffer, Judith; Schmidt, Jan C. (Hrsg.): Perspektiven interdisziplinärer Technikforschung. Konzepte, Analysen, Erfahrungen. Münster: agenda-Verlag, 2002a, S. 209-219.

[Bi02b] Bittner, Peter: Unser aller Profession gib uns heute ... oder die Frage nach einer mäeutischen Informatik. In: Nake, Frieder; Rolf, Arno; Siefkes, Dirk (Hrsg): Wozu Informatik? Theorie zwischen Ideologie, Utopie und Phantasie. Materialien zu einer Arbeitstagung in Bad Hersfeld März 2002. Berlin: TU Berlin [Forschungsberichte der Fakultät IV - Elektrotechnik und Informatik, Bd.: 2002-25], S. 42-45.

[Bi03a] Bittner, Peter: Theorien der Informatik – allgemein, handlungsorientiert, mäeutisch. Ein „kritisches" Manifest. Beitrag zu einem Sammelband anläßlich der Emeritierung von Prof. Dr. Rudolf Wille. 2003a. (erscheint im Verlag Allgemeine Wissenschaft).

[Bi03b] Bittner, Peter: Informatik (anders) denken ... Über „gute" Disziplinarität, Kritische Theorie und Informatik. In: Böhme, Gernot; Manzei, Alexandra (Hrsg.): Kritische Theorie der Natur und der Technik. München: Wilhelm Fink Verlag, 2003b (in Vorbereitung).

[CW33] Carr-Saunders, Alexander M.; Wilson, Paul A.: The Professions. Oxford University Press, 1933 (reprinted London: Frank Cass, 1964).

[De01] Denning, Peter J.: Who are we? Communications of the ACM (CACM), 44 (2), 2001, S. 15-19.

[FG96] Ford, Gary; Gibbs, Norman E.: A Mature Profession of Software Engineering. Technical Report, CMU/SEI-96-TR-004 ESC-TR-96-004, Carnegie Mellon University: Software Eng. Institute, September 1996.

[Go57] Goode, William J.: Community within a Community: The Professions. American Sociological Review, 22, 1957, S. 194-200.

[Go72] Goode, William J.: Professionen und die Gesellschaft. Die Struktur ihrer Beziehungen. In: Luckmann, Thomas; Sprondel, Walter M. (Hrsg.): Berufssoziologie. Köln: Kiepenheuer & Witsch, 1972, S. 157-167.

[Ha95] Hartmann, Michael: Informatiker in der Wirtschaft. Perspektiven eines Berufs. Berlin u.a.: Springer, 1995.

[He72] Hentig, Hartmut von: Magier oder Magister? Über die Einheit der Wissenschaft im Verständigungsprozeß. Stuttgart: Klett, 1972.

[Ho02] Hofer, Christian: Die Beratungskomponente in der Softwareentwicklung im Spannungsfeld von technischer Problemlösung und stellvertretender Krisenbewältigung. Exemplarische Fallanalysen mit dem Verfahren der objektiven Hermeneutik. Frankfurt: JWG-Universität, Diplomarbeit am Fachbereich Gesellschaftswissenschaften, 2002.

[HB00] Hornecker, Eva; Bittner, Peter: Vom kritischen Verhältnis zur Berufspraxis in der Informatik – Ergebnisse einer Befragung. FIfF-Kommunikation 1/2000, S. 33-39.

[Ko96] Koring, Bernhard: Zur Professionalisierung der pädagogischen Tätigkeit. Beiträge aus erziehungs- und sozialwissenschaftlicher Sicht. In: Combe, Arno; Helsper, Werner (Hrsg.): Pädagogische Professionalität. Untersuchungen zum Typus pädagogischen Handelns. Frankfurt/Main: Suhrkamp, 1996, S. 303-339.

[Ko99] Koring, Bernhard: Grundprobleme pädagogischer Berufstätigkeit. Thema 6: Die Frage nach der Professionalität pädagogischer Tätigkeit. (Text und Folien zur Veranstaltung) http://www-user.tu-chemnitz.de/~koring/sem-vl-paed-beruf/tma6.htm Stand: 05.10.1999 (gesehen am 06.01.2003).

[No02] Nohl, Herman: Die pädagogische Bewegung in Deutschland und ihre Theorie. Frankfurt/Main: Vittorio Klostermann, [11]2002 (zuerst 1933).

[Oe78] Oevermann, Ulrich: Probleme der Professionalisierung in der berufsmäßigen Anwendung sozialwissenschaftlichen Kompetenz. Frankfurt/Main: unveröffentlichtes Manuskript, 1978.

[Oe83] Oevermann, Ulrich: Hermeneutische Sinnrekonstruktion: Als Therapie und Pädagogik mißverstanden, oder: das notorische strukturtheoretische Defizit pädagogischer Wissenschaft. In: Garz, Detlev; Kraimer, Klaus: Brauchen wir andere Forschungsmethoden? Frankfurt/Main: Scriptor, 1983, S. 113-155.

[Oe96] Oevermann, Ulrich: Theoretische Skizze einer revidierten Theorie professionalisierten Handelns. In: Combe, Arno; Helsper, Werner (Hrsg.): Pädagogische Professionalität. Untersuchungen zum Typus pädagogischen Handelns. Frankfurt/Main: Suhrkamp, 1996, S. 70-182.

[Pa39] Parsons, Talcott: The Professions and Social Structure. Social Forces, 17, 1939, S. 457-467.

[SK01] Schinzel, Britta; Kleinn, Karin: Quo vadis, Informatik? Informatik-Spektrum, 24 (2), 2001, S. 91-97.

[St92] Stichweh, Rudolf: Professionalisierung, Ausdifferenzierung von Funktionssystemen, Inklusion. Betrachtungen aus systemtheoretischer Sicht. In: Dewe, Bernd; Ferchhoff, Wilfried; Radtke, Frank-Olaf (Hrsg.): Erziehen als Profession. Zur Logik professionellen Handelns in pädagogischen Feldern. Opladen: Leske + Budrich, 1992, S. 36-48.

[Wa96] Waddington, I.: Professions. In: Kuper, Adam; Kuper, Jessica (eds.): The Social Science Encyclopedia. London: Routledge, ^{2}1996, S. 677-678.

[Wi88] Wille, Rudolf: Allgemeine Wissenschaft als Wissenschaft für die Allgemeinheit. In: Böhme, Helmut; Gamm, Hans-Jochen (Hrsg.): Verantwortung in der Wissenschaft. Darmstadt: TH Darmstadt (THD-Schriftenreihe Wissenschaft und Technik; Bd. 43), 1988, S. 159-176. Nachdruck in: Conceptus – Zeitschrift f. Philosophie, 60, S. 117-128.

[Wi96] Wille, Rudolf: Allgemeine Mathematik – Mathematik für die Allgemeinheit. TH Darmstadt: FB4-Preprint Nr. 1822, 1996.